MA6150. Algoritmos de Aproximación. 2023.
Profesor: José Soto
Última Actualización: 30 de noviembre de 2023.

Ingeniería Matemática
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Guía de ejercicios.

Esta guía se irá actualizando con ejercicios para el curso. En cada tarea se pedirá entregar cierto puntaje mínimo de ellos, de ciertas secciones. Se podría solicitar entregar algún problema específico adicionalmente. Se recomienda trabajar y resolver ejercicios de la guía de manera sistemática.

1. Generales

En todos los algoritmos que proponga debe justificar el factor de aproximación objenido.

Ejercicio 1.1 (1 punto)**.** En el problema MÁXIMO SUBDIGRAFO ACÍCLICO, la entrada es un digrafo $G = (V, E)$ y la salida debe ser un conjunto de arcos $F \subseteq E$ acíclico (en el sentido dirigido) de cardinalidad máxima.

(a) Encuentre una 2-aproximación a este problema. **Indicación:** Enumere $V = [n]$ y llame a un arco $e = (i, j) \in E$ *directo* si $i < j$ o *reverso* si $j < i$. Estudie ambos conjuntos.

(b) Suponga ahora que el grafo $\overline{G}$ no dirigido subyacente de G es conexo y simple. Llame $\rho = |E|/(|V| - 1)$ a la densidad de G. Encuentre una $\min(\rho, 2)$-aproximación para el problema del *maximo subdigrafo acíclico*.

Ejercicio 1.2 (1 punto)**.**

(a) En el problema del MÍNIMO MATCHING MAXIMAL (MMM) el objetivo es encontrar un matching maximal M de cardinalidad mínima. Demuestre que el algoritmo greedy que encuentra cualquier matching maximal es una 2-aproximación para este problema.

(b) Considere ahora el mismo problema anterior pero en un hipergrafo k-uniforme. ¿Es o no cierto que el mismo algoritmo greedy es una k-aproximación para MMM? (justifique o dé un contraejemplo)

Ejercicio 1.3. (1 punto) Sea $k \geq 1$ cualquier constante. Demuestre que si **P** $\neq$ **NP** no existe algoritmo para MIN VERTEX COVER que devuelva un vertex cover W con garantía aditiva de k, es decir, tal que $|W| \leq \text{OPT} + k$.

Ejercicio 1.4 (1 punto)**.**
Encuentre una $O(n/\log n)$ aproximación para MAX CLIQUE siguiendo el siguiente esquema: Para $k = k(n)$ cierta función de n, elija una partición cualquiera de V en grupos de tamaño k. Revisar *todos* los subconjuntos de V completamente contenidos en alguno de los grupos y devolver de todos ellos el clique más grande. Primero, demuestre que si $k(n) = O(\log n)$ entonces el algoritmo anterior se puede implementar en tiempo polinomial. Luego demuestre que la solución obtenida es una $n/k(n)$ aproximación. Finalmente use $k(n) = \log n$ para concluir.

Ejercicio 1.5. (1 punto) Considere el siguiente algoritmo para Min Vertex Cover

Algoritmo 1: Para encontrar un vertex-cover.

Entrada: Un grafo G.
Salida: Un vertex cover ALG

```
ALG ← ∅.
G' ← G
mientras E(G') ≠ ∅ hacer
    Sea v ∈ V(G') el vértice de mayor grado de G'.
    Añadir v a ALG.
    Borrar de G', el vértice v y todas las aristas incidentes a v.
fin
devolver ALG.
```

(a) Muestre que el algoritmo anterior es correcto (devuelve un vertex cover en tiempo polinomial).

(b) Muestre una familia infinita de instancias donde el algoritmo anterior devuelva una solución ALG con alg $\geq \Omega(\log n)$opt. **Indicación:** Considere el grafo G_n bipartito con lado izquierdo $[n]$ y lado derecho $\mathcal{P}([n])$, donde para cada $j \in [n]$ y $A \in \mathcal{P}([n])$, jA es arista si y solo si $j \in A$. ¿Qué podría pasar en este grafo?

Ejercicio 1.6. (1 punto) El problema Max Cut se define como sigue. Dado un grafo G, encontrar un conjunto $S \subseteq V(G)$ (un corte) tal que el número de aristas con un extremo en S y otro en $V(G) \setminus S$ se maximice, es decir estamos maximizando $|\delta(S)| = |E(S : V(G) \setminus S)|$. Considere el siguiente algoritmo para Max Cut.

Algoritmo 2: Para encontrar un corte.

Entrada: Un grafo G.
Salida: Un corte ALG

```
ALG ← ∅.
mientras verdadero hacer
    si Existe v ∈ V(G) \ ALG tal que |δ(ALG + v)| > |δ(ALG)| entonces
        ALG ← ALG + v.
    si no, si Existe v ∈ ALG tal que |δ(ALG − v)| > |δ(ALG)| entonces
        ALG ← ALG − v.
    en otro caso
        devolver ALG.
    fin
fin
```

Muestre que el algoritmo anterior es correcto (devuelve un corte en tiempo polinomial) y que es una 2-aproximación para MAX CUT.

2. Algoritmos combinatoriales con desigualdad triangular

Ejercicio 2.1 (3 puntos)**.** Resulta útil conocer algunos algoritmos exactos de b-matchings, b-matchings simples y b-factores. Sea G un grafo, $b \in \mathbb{Z}^V$ una función de grados en los vértices y $w \in \mathbb{R}^E$ una función de pesos en las aristas. Un b-matching es una asignación de valores $x(e) \in \{0, 1, 2, \ldots, b\}$ a cada arista e de un grafo de modo que para cada vértice v, $x(\delta(v)) \leq b_v$. (Se pude interpretar como que estamos tomando $x(e)$ copias

de la arista e) Un b-matching perfecto es un b-matching donde $x(\delta(v)) = b_v$. Y un b-factor es un b-matching perfecto tal que $x(e) \in \{0, 1\}$ para cada arista. Así por ejemplo, los 1-matching son exactamente los matchings y los 1-factores los matchings perfectos.

En la literatura existen algoritmos polinomiales para calcular b-matchings y b-matchings perfectos de peso máximo/mínimo, y también para calcular b-factores de peso mínimo. En este ejercicio nos dedicaremos a diseñar algoritmos para casos particulares.

(a) Pruebe que si G es bipartito entonces se pueden resolver los 3 problemas (b-matching de peso máximo, b-matching perfecto de peso máximo/mínimo y b-factores de peso mínimo) con programación lineal (**Indicación**: use total unimodularidad).

(b) Sea ahora G un grafo general y suponga que b es una función a valores pares (pero arbitrarios). Sea $V' = \{v' : v \in V\}$ una copia de V. Considere el grafo bipartito H con bipartición V y V' tal que para todo $uv \in E(G)$ hay 2 aristas uv' y vu' en $E(H)$. Defina además grados $\tilde{b}_v = \tilde{b}_{v'} = b_v/2$ para cada $v \in V$. Demuestre que encontrar un b-matching (normal o perfecto) de peso máximo es equivalente a encontrar un $\tilde{b}$-matching (normal o perfecto, respectivamente) de peso máximo en H.

(c) Sea G un grafo general, b una función a valores enteros acotados por algún polinomio en n (digamos $b_v \leq n^{O(1)}$). Reduzca el problema de encontrar un b-matching (normal o perfecto) de peso máximo al problema de encontrar un matching (normal o perfecto) de peso máximo en un grafo H mucho más grande (pero con un número polinomial en $|G|$ de vértices y aristas). **Indicación:** Use b_v copias de cada v, asigne aristas y pesos de manera adecuada.

(d) Sea G un grafo general, b una función a valores arbitrarios tal que G admite b-factores. Reduzca el problema de encontrar un b-factor de peso mínimo al problema de encontrar un b-matching perfecto de peso máximo en un grafo auxiliar H. **Indicación:** Subdivida cada arista $e = uv$ de E en un camino de 3 aristas (digamos ue_u, e_ue_v, e_vv). Asigne capacidades b' apropiadas a los $|V| + 2|E|$ vértices, y pesos a las aristas de modo tal que todo b'-matching perfecto de peso máximo en H corresponda a un b-factor de peso mínimo en G

Ejercicio 2.2 (1 punto)**.** Un cycle-cover (disjunto) de un grafo dirigido es un subgrafo F formado por colección de ciclos nodo disjuntos que cubre cada nodo (es decir, el grado de entrada y el grado de salida de cada nodo en F es 1).

Supongamos ahora que G es un digrafo completo. Diseñe un algoritmo polinomial para el problema de encontrar un cycle-cover de peso mínimo. **Indicación:** Redúzcase al problema de encontrar un matching perfecto de peso mínimo en un grafo bipartito (lo que es polinomial).

Ejercicio 2.3 (1 punto)**.** Sea G un digrafo completo y $\ell : E(G) \to \mathbb{R}_+$ una función de distancia métrica asimétrica. Sea opt el valor de un ATSP óptimo en G.

Suponga que recibe un sub(multidi)grafo H de G que es Euleriano (fuertemente conexto, con grado de entrada igual a salida en cada nodo) tal que $\ell(H) \leq \alpha\text{opt}$. Diseñe un algoritmo polynomial que entregue, a partir de H, un tour T de G con $\ell(T) \leq (\alpha + \beta)\text{opt} = (\alpha + O(1))\text{opt}$. ¿Cuán pequeño puede hacer β?

Ejercicio 2.4 (2 puntos)**.** En el problema del TSP-path métrico, la entrada es un grafo completo G con largos métricos en sus aristas. La salida es determinar un camino que pase por todos los vértices de largo total mínimo. Usando las mismas ideas del algoritmo de Christofides encuentre una 3/2-aproximación para este problema.

El problema es más complejo si además se especifican los extremos s y t del camino. Rehaga el análisis mostrando que el algoritmo es en verdad una 5/3-aproximación para el s-t TSP-path métrico.

Indicación: Llame H al multiconjunto obtenido al unir las aristas del MST T y de OPT juntos. Llame S al conjunto de vértices de grado equivocado en T y M al matching de costo mínimo entre vértices de S de su algoritmo. Demuestre que H se puede particionar en un conjunto H_1 consistente de caminos arista-disjuntos con extremos en S y un ciclo Euleriano H_2. Use esto para deducir que $\ell(H) \geq 3\ell(M)$ y concluya.

Ejercicio 2.5 (2 puntos)**.** Sea G un grafo completo y $\ell\colon E \to \{1, 2\}$ una asignación de largos tal que cada arista tiene valor 1 o 2. Es claro que entonces ℓ satisface desigual triangular. Encuentre una aproximación mejor que 3/2 para el problema del TSP métrico en (G, ℓ). ¿Qué factor puede obtener?

Indicación: Del ejercicio 2.1 se puede encontrar un 2-factor H de peso mínimo en tiempo polinomial. Diseñe un método iterativo para *pegar* todos los ciclos de H en uno solo aumentando el largo total en a lo más en el número de ciclos de H.

Ejercicio 2.6 (1 punto)**.** Sea G un grafo conexo y $\ell\colon E \to \mathbb{R}_+$ no necesariamente métrico.

En el k-Steiner Tree, se busca encontrar un Steiner Tree de largo mínimo que use no más de k vértices de Steiner. Pruebe que si $k = O(\log n)$, entonces el k-Steiner Tree se puede resolver en tiempo polinomial (en el tamaño del grafo).

Ejercicio 2.7 (2 puntos)**.** Sea G un grafo completo y $\ell\colon E \to \mathbb{R}_+$ distancias métricas y $k \geq 0$. En el problema del k-cluster mínimo métrico el objetivo es particionar V en k conjuntos $V_1, \ldots, V_k$ de modo de minimizar $\max_i \operatorname{diam}(V_i)$, donde $\operatorname{diam}(X)$ es el diámetro de X es decir, la mayor distancia entre pares de puntos de X.

Diseñe una 2-aproximación para k-cluster mínimo métrico.

Indicación: El algoritmo debería ser muy similar al de k-center pero debe preocuparse de construir los clusters. Pruebe el siguiente resultado. Si existe $X \subseteq V$ con $|X| = k+1$ tal que todo par de puntos de X están a distancia al menos h entonces opt $\geq h$.

Ejercicio 2.8 (2 puntos)**.** Sea G un grafo completo, $\ell\colon E \to \mathbb{R}_+$ distancias métricas y $k \geq 0$. Adicionalmente V está particionado en clientes C y proveedores P.

En el problema de los k-suppliers métrico, el objetivo es encontrar un subconjunto $Q \subseteq P$ de exactamente k proveedores tal que la máxima distancia de un cliente a un proveedor se minimice. En otras palabras se busca minimizar $\max_{j \in C} \min_{i \in Q} \ell(j, i)$. Diseñe una 3-aproximación para k-suppliers.

Indicación: El problema en principio es similar al del ejercicio anterior. Use el algoritmo de k-center sobre C para encontrar k clientes especiales. Defina a lo más k proveedores a partir de ellos.

Ejercicio 2.9. (1 punto)

Un grafo de Barnette es un grafo cúbico, 3 vértice-conexo, bipartito y planar. Se conjetura (no puede usar este hecho) que estos grafos son Hamiltonianos. Demuestre que si G es de Barnette entonces su completación métrica admite un tour Hamiltoniano de largo a lo más $4n/3$. Para esto puede usar sin demostrar que cuando G se dibuja en el plano, resulta ser 3-cara coloreable y que dicho coloreo se puede encontrar en tiempo polinomial.

Indicación: Demuestre, usando propiedades de grafos planares y bipartitos, que alguno de los 3 cycle-covers inducidos por los colores de las caras de G tiene pocos ciclos.

Ejercicio 2.10. (1 punto)

Considere la relajación natural para Steiner Tree como PL, donde R es el conjunto de terminales.

$$\begin{aligned}(P)\,\text{mín}\ & \sum_{e\in E}\ell_e x_e \\ \text{s.a.}\quad & x(\delta(S))\geq 1 \quad \forall S\subseteq V, |S\cap R|\geq 1, |R\setminus S|\geq 1, \\ & x\geq 0\end{aligned}$$

Demuestre que (P) es efectivamente una relajación y que su gap de integralidad es al menos 2, incluso en el caso de MST (es decir, cuando $R = V$). **Indicación:** ¿Qué pasa en un ciclo?

Ejercicio 2.11. (1 punto)

Demuestre que el gap de integralidad de la relajación de Held-Karp para TSP métrico es al menos 4/3.

Indicación: Considere la instancia gráfica dada por dos triángulos conectados por 3 caminos con k vértices cada uno. Necesita calcular el valor de opt, proponer un vector x fraccional en el poliedro (argumentando por qué lo está) y demostrar que opt $\geq (4/3)\ell(x)$.

Ejercicio 2.12. (2 punto)

Demuestre que el gap de integralidad de la relajación de Held-Karp para TSP métrico es a lo más 3/2. Para esto, basta demostrar que el algoritmo de Christofides devuelve un tour de valor a lo más $3/2\text{opt}_{PL}$ ¿vé por qué?

En este problema puede usar que, para cada grafo G, siguientes poliedros son integrales, sin demostrarlo.

(I) $P_1 = \text{conv}(\chi^Q\colon Q \text{ árbol generador de } G) = \{x\in\mathbb{R}^E\colon x(\delta(S))\geq 1, \forall\emptyset\subsetneq S\subsetneq 1, x(E) = n-1, x\geq 0\}$

(II) Para todo $T\subseteq V$, un conjunto $J\subseteq E$ es un T-join si T es exactamente el conjunto de vértices de grado impar en (V, J).
$P_2(T) = \text{conv}(\chi^J\colon J \text{ es un } T\text{-join de } G) = \{x\in\mathbb{R}^E\colon x(\delta(S))\geq 1, \forall\emptyset\subsetneq S\subsetneq 1 \text{ con } |S\cap T| \text{ impar }, x\geq 0\}$

Llame x^* a un punto fraccional óptimo de la relajación de Held-Karp, llame Q y M al árbol generador y matching encontrado por Christofides. Pruebe que $x^*\cdot(n-1)/n\in P_1$, $x^*\cdot 1/2\in P_2(T)$ y deduzca que $\ell(Q)+\ell(M)\leq(3/2)\ell(x^*)$.

3. Algoritmo greedy y sistemas de independencia

Ejercicio 3.1 (1 punto)**.**

(a) Demuestre formalmente que todo sistema k-intercambiable es un k-sistema.

(b) Encuentre un 2-sistema (con pocos elementos) que no sea 2-intercambiable.

Ejercicio 3.2 (1 punto)**.** Demuestre que el sistema de matchings del grafo $C_5 = ([5], E)$, (ciclo en 5 vértices) no puede ser descrito como la intersección de 2 matroides.
Indicación: Estudie los circuitos (no-independientes minimales) de tamaño 2 del sistema de matching. ¿Qué pasa si alguna de las matroides contiene 3 de esos circuitos? Use propiedades de los circuitos de matroides.

Ejercicio 3.3 (1 puntos)**.**

(a) Demuestre que si $(S, \mathcal{I}_1)$ es c_1-intercambiable y $(S, \mathcal{I}_2)$ es c_2-intercambiable entonces $(S, \mathcal{I}_1 \cap \mathcal{I}_2)$ es $(c_1 + c_2)$-intercambiable.

(b) Demuestre que si $(S, \mathcal{I}_1)$ es un c_1-sistema y $(S, \mathcal{I}_2)$ es un c_2-sistema entonces $(S, \mathcal{I}_1 \cap \mathcal{I}_2)$ es un $(c_1 + c_2)$-sistema.

Ejercicio 3.4 (1 punto)**.** Sea k entero positivo. Un sistema de independencia $(S, \mathcal{I})$ es una k-matchoide si existen $m \geq 0$ matroides $(S_1, \mathcal{I}_1)$, $(S_2, \mathcal{I}_2)$, $\ldots$, $(S_m, \mathcal{I})$ tal que cada elemento de S pertenece a lo más k conjuntos S_i, y además $\mathcal{I} = \{X \subseteq S \colon \forall i \in [m], X \cap S_i \in \mathcal{I}_i\}$. Demuestre que las k-matchoides son k-intercambiables.

Ejercicio 3.5 (1 punto)**.** Sea $(N, \mathcal{I})$ un sistema de independencia sin loops (un loop es un elemento $e \in N$, con $\{e\} \notin \mathcal{I}$) y $w \colon S \to \mathbb{R}_+$. Es posible implementar el algoritmo greedy *procesando* los elementos de N de mayor a menor peso. Resulta interesante intentar modificar el algoritmo de modo que los elementos puedan presentarse de manera *online*, en un orden adversarial. Considere el siguiente algoritmo alternativo.

Algoritmo 3: Greedy vía intercambios

Entrada: Un sistema k-intercambiable $(N, \mathcal{I})$ sin *loops*, y una función de peso $w \colon S \to \mathbb{R}_+$ estrictamente positiva.

```
Sea e_1, ..., e_s un ordenamiento arbitrario de N.
ALG ← ∅.
para i desde 1 hasta s hacer
    Determinar el conjunto X ⊆ ALG, |X| ≤ k tal que ALG + e_i \ X ∈ I con menor w(X).
    si w(e_i) > w(X) entonces ALG ← ALG + e_i \ X;
fin
devolver ALG.
```

(a) Pruebe que si $k = 1$ el algoritmo anterior es exacto para determinar un independiente de peso máximo.

(b) Refute que si $k \geq 2$ el algoritmo anterior es una k-aproximación (piense un contraejemplo para $k = 2$ donde el algoritmo anterior sea pésimo).

Ejercicio 3.6 (1 punto)**.** Sea $G = (V, E)$ un grafo. Llame

$$\mathcal{I} = \{F \subseteq E \colon \text{ cada componente conexa de } (V, F) \text{ contiene a lo más un ciclo}\}.$$

Pruebe directamente usando alguna caracterización axiomática de matroides (o 1-sistemas, 1-intercambiables) que $(E, \mathcal{I})$ es una matroide. **Nota:** Estas matroides se conocen como matroides bicirculares.

Ejercicio 3.7 (1 punto)**.** Sea $(S, \mathcal{I})$ un sistema de independencia. Decimos que es c-extendible si

$$\forall C \subseteq D \in \mathcal{I}, \forall x \in S \setminus D, \qquad (C + x) \in \mathcal{I} \implies \exists Y \subseteq D \setminus C, |Y| \leq c, \text{tal que } D \setminus Y + x \in \mathcal{I}.$$

Demuestre que un sistema es c-extendible si y solo si es c-intercambiable.

Ejercicio 3.8 (1 punto)**.** Sea $G = (V, E)$ un grafo conexo y $w\colon E \to \mathbb{R}_+$ una función de peso. Considere el conjunto $\mathcal{I} = \{M \subseteq E\colon M \text{ es un matching y } G \setminus M \text{ es conexo}\}$. Llame A al algoritmo greedy para determinar el objeto $M \in \mathcal{I}$ de peso máximo. Demuestre que A es siempre una 3-aproximación pero no es una 2-aproximación en general.

4. Funciones submodulares

Ejercicio 4.1 (1 punto)**.** Complete las siguientes demostraciones.

(a) Probar que si $f\colon 2^S \to \mathbb{R}$ satisface que $\forall A \subseteq B \subseteq S, \quad f(B) \leq f(A) + \sum_{x \in B \setminus A} f_A(x)$ entonces f es submodular.

(b) Probar que si $f\colon 2^S \to \mathbb{R}$ satisface que $\forall A \subseteq S, B \subseteq S, \quad f(B) \leq f(A) + \sum_{x \in B \setminus A} f_A(x)$ entonces f es submodular monótona.

Ejercicio 4.2 (1 punto)**.** Una función $f\colon 2^S \to \mathbb{R}$ se dice *supermodular* si $-f$ es submodular. La función se dice *modular* si es submodular y supermodular. Determine (con demostración) si las siguientes funciones son modulares, submodulares o supermodulares:

- La función $cc\colon 2^E \to \mathbb{R}$, donde $cc(X) =$ número de componentes conexas del grafo (V, X) donde $G = (V, E)$ es un grafo dado.
- Dado un conjunto finito de eventos $S = \{A_1, A_2, \ldots, A_k\}$ en un espacio de probabilidad cualquiera, estudie la función $f\colon 2^{[k]} \to \mathbb{R}$ dada por $f(X) = \Pr(\bigwedge_{i \in X} A_i)$.

Ejercicio 4.3 (2 puntos)**.** Extensión Multilineal. Toda función $f\colon 2^S \to \mathbb{R}$, se puede interpretar como una función con dominio los vértices del hipercubo $\{0, 1\}^S$, interpretando $f(\chi^X)$ como $f(X)$. Un ingrediente fundamental para (los actualmente mejores) algoritmos para maximizar funciones submodulares sobre familias de conjuntos, consiste en usar métodos para extender f a una función F con dominio el hipercubo completo $[0, 1]^S$ de modo tal que F tenga buenas propiedades. Para simplificar notación supondremos que $S = [n]$.

Definimos la extensión multilinear F de f como $F\colon [0, 1]^n \to \mathbb{R}$, donde $F(x) = \mathbb{E}[f(X)]$ donde $X \subseteq [n]$ es el conjunto aleatorio obtenido al fijar $i \in X$ con probabilidad x_i de manera independiente. Es decir,

$$F(x) = \sum_{X \subseteq S} f(X) \prod_{i \in X} x_i \prod_{i \in S \setminus X} (1 - x_i).$$

Notamos que para todo $X \subseteq S$, $F(\chi^X) = f(X)$.

(a) Demuestre que f es monótona si y solo si $\frac{\partial F}{\partial x_i}(x_0) \geq 0$ para todo $x_0 \in (0, 1)^n$, para todo $i \in [n]$

(b) Concluya que si f es monótona entonces F es no-decreciente en cualquier dirección $d \geq 0$ (es decir, pruebe que para todo $x_0 \in (0, 1)^n, F(x_0 + td)$ es no-decreciente en t).

(c) Demuestre que f es submodular si y solo si $\frac{\partial^2 F}{\partial x_i x_j}(x_0) \leq 0$ para todo $x_0 \in (0,1)^n$, $i, j \in [n]$.

(d) Concluya que si f es submodular entonces F es cóncava en cualquier dirección $d \geq 0$ (es decir, $F(x_0 + td)$ es cóncava), y que F es convexa en la dirección $d = e_i - e_j$ para todo $i, j \in N, i \neq j$.

Ejercicio 4.4. (2 puntos) En este ejercicio veremos un algoritmo para maximizar funciones submodulares no sean necesariamente normalizadas o monótonas.

Considere el siguiente algoritmo doble-greedy para maximizar una función submodular no negativa.

Algoritmo 4: Doble-Greedy

Entrada: Una función $f\colon 2^{[n]} \to \mathbb{R}$

$X_0 \leftarrow \emptyset, Y_0 \leftarrow [n]$
para $i = 1$ *hasta* n **hacer**
  $a_i \leftarrow f(X_{i-1} + i) - f(X_{i-1})$
  $b_i \leftarrow f(Y_{i-1} - i) - f(Y_{i-1})$
  si $a_i \geq b_i$ **entonces**
    $X_i \leftarrow X_{i-1} + i$, $Y_i \leftarrow Y_{i-1}$
  en otro caso
    $X_i \leftarrow X_{i-1}$, $Y_i \leftarrow Y_{i-1} - i$
  fin
fin
devolver X_n.

Sea $f\colon 2^{[n]} \to \mathbb{R}$ una función submodular no negativa (no necesariamente monótona ni normalizada). Apliquemos el algoritmo doble-greedy a f. Sea OPT un conjunto que maximiza f, $\overline{\text{OPT}} = [n] \setminus \text{OPT}$ y defina para cada i, $\text{OPT}_i = (\text{OPT} \cup X_i) \cap Y_i$, $\overline{\text{OPT}}_i = (\overline{\text{OPT}} \cup X_i) \cap Y_i$. Pruebe que para todo $i \in [n]$

(a) $a_i + b_i \geq 0$.

(b) $\text{OPT}_i \cap [i] = X_i \cap [i] = Y_i \cap [i] = \overline{\text{OPT}}_i$. En particular $\text{OPT}_n = X_n = Y_n = \overline{\text{OPT}}_n$.

(c) $\text{OPT}_i \setminus [i] = \text{OPT} \setminus [i]$, $\overline{\text{OPT}}_i \setminus [i] = \overline{\text{OPT}} \setminus i$. En particular $\text{OPT}_0 = \text{OPT}$, $\overline{\text{OPT}}_0 = \overline{\text{OPT}}$.

(d)

$$f(\text{OPT}_{i-1}) - f(\text{OPT}_i) \leq (f(X_i) - f(X_{i-1})) + (f(Y_i) - f(Y_{i-1})).$$
$$f(\text{OPT}_{i-1}) - f(\text{OPT}_i) + f(\overline{\text{OPT}}_{i-1}) - f(\overline{\text{OPT}}_i) \leq (f(X_i) - f(X_{i-1})) + (f(Y_i) - f(Y_{i-1}))$$

Hint: Pruebe ambas desigualdades simultáneamente, considerando 4 casos: de acuerdo a si $a_i \geq b_i$ o no, y a si $i \in \text{OPT}$ o no.

Usando un argumento telescópico, concluya que el algoritmo doble-greedy es una 3-aproximación para maximizar funciones submodulares no negativas, y es una 2-aproximación para maximizar funciones submodulares no negativas simétricas (una función se dice simétrica si $f(X) = f(S \setminus X)$, un ejemplo de estas funciones es la función de corte de un grafo con pesos).

Ejercicio 4.5. (2 puntos) En este problema probaremos que greedy es una (c+1)-aproximación para maximizar una función $f\colon 2^S \to \mathbb{R}_+$ SMN sobre c-sistemas sin loops (esto es, asumimos que $\{x\} \in \mathcal{I}$ para todo $x \in S$, esto se puede suponer sin pérdida de generalidad eliminando los loops de S primero).

Solo para el análisis modificaremos ligeramente la descripción del algoritmo.

Algoritmo 5: Greedy

Entrada: Una función $f\colon 2^S \to \mathbb{R}$, un c-sistema $(S, \mathcal{I})$.

```
ALG ← ∅, N ← S i ← 0. U_0 ← ∅. mientras N ≠ ∅ hacer
    Elegir u ∈ N que maximice f_ALG(u).
    N ← N − u si ALG + u ∉ I entonces
        U_i ← U_i + u
    en otro caso
        U_{i+1} ← {u}
        ALG ← ALG + u
        i ← i + 1
    fin
fin
devolver ALG.
```

Llamemos $\text{ALG} = \{a_1, \ldots, a_k\}$ a los elementos de ALG en el orden que fueron añadidos, $A_j = \{a_1, \ldots, a_j\}$. De acuerdo a la implementación anterior (y del hecho que no hay loops) notamos que al final del algoritmo $U_0 = 0$, y U_i son todos los elementos considerados entre a_i (incluido) hasta justo antes de añadir a_{i+1} (o terminar, en el caso $i = k$). En particular $S = \bigcup_{i=1}^{k} U_i$. Llame además $s_i = |\text{OPT} \cap U_i|$.

(a) Demuestre que $\displaystyle\sum_{x \in \text{OPT} \setminus \text{ALG}} f_{\text{ALG}}(x) \leq \sum_{i=1}^{k} s_i f_{A_{i-1}}(a_i)$

(b) Demuestre que para todo $j \leq k$, $\sum_{i=1}^{j} s_i \leq cj$.

(c) Sea $(\rho_i)_{i \in [k]}$ una secuencia decreciente y no-negativa. Pruebe que si $(y_i)_{i \in [k]}$, es una secuencia de valores no negativos tal que para todo $j \leq k$, $\sum_{i=1}^{j} y_i \leq j$, entonces se debe cumplir que $\sum_{i=1}^{k} y_i \rho_i \leq \sum_{i=1}^{k} \rho_i$. **Hint:** Escriba un programa lineal en variables y_i que calcule el máximo valor posible de $\sum_{i=1}^{k} \rho_i y_i$. Use su dual para probar que $\sum_{i=1}^{k} i(\rho_i - \rho_{i+1})$ es cota superior (considerando $\rho_{k+1} = 0$),

(d) Use las partes (b) y (c) anteriores para deducir que $\sum_{i=1}^{k} s_i f_{A_{i-1}(a_i)} \leq cf(\text{ALG})$.

(e) Concluya que ALG es una $(c+1)$-aproximación.

Ejercicio 4.6. (2 puntos) Considere el problema de maximizar una función submodular no-negativa $f\colon 2^S \to \mathbb{R}$ no necesariamente monótona, sobre una matroide uniforme de rango k (i.e., sobre $(S, \mathcal{I})$ con $\mathcal{I} = \{X \subseteq S\colon |X| \leq k\}$) Por simplicidad, supondremos que $|S| \geq k$, considere la siguiente versión aleatoria del algoritmo greedy (ver algoritmo Random Greedy).

Indicación: Para las siguientes partes, conviene condicionar en la identidad del conjunto A_{i-1} y luego descondicionar.

(a) Pruebe que para todo $i \in [k]$, $\mathbb{E}[f(A_i \cup \text{OPT})] \geq (1 - 1/k)^i f(\text{OPT})$.

(b) Pruebe que para todo $i \in [k]$, $k\mathbb{E}[f_{A_{i-1}}(u_i)] \geq \mathbb{E}[f_{A_{i-1}}(\text{OPT})] \geq (1 - 1/k)^{i-1} f(\text{OPT}) - \mathbb{E}[f(A_{i-1})]$

(c) Concluya que para todo $i \in [k]$, $\mathbb{E}[f(A_i)] \geq \frac{i}{k}\left(1 - \frac{1}{k}\right)^{i-1} f(\text{OPT})$, y deduzca de aquí que el algoritmo es una e-aproximación en esperanza.

Algoritmo 6: Random Greedy

Entrada: f, k

$A_0 \leftarrow \emptyset$
para *i desde 1 hasta k* **hacer**
  Determinar el conjunto M_i que contiene a los k elementos x de $S \setminus A_{i-1}$ de mayor valor $f_{A_{i-1}}(x)$.
  Sea $u_i \in M_i$ un elemento elegido uniformemente al azar.
  $A_i \leftarrow A_{i-1} + u_i$.
fin
Devolver A_k.

Ejercicio 4.7. (3 puntos) Sea $c\colon S \to \mathbb{R}_+$ una función de costo y B un presupuesto positivo. Sea además f una función SMN sobre S. Un conjunto $X \subseteq S$ se dice factible para el problema de la mochila si $c(X) \leq B$. Considere el problema de encontrar el conjunto factible X que maximiza $f(X)$. Considere el siguiente algoritmo greedy que recibe un conjunto factible Q como entrada y que en cada momento agrega a la solución aquel elemento que maximiza su contribución marginal dividido por su costo.

Algoritmo 7: Greedy para knapsack.

ALG $\leftarrow \emptyset$.
$\hat{S} = S$
mientras $\hat{S} \neq \emptyset$ **hacer**
  $x^* \leftarrow \arg\max_{x \in \hat{S}} \frac{f_{\text{ALG}}(x)}{c(x)}$
  si $c(\text{ALG}) + c(x^*) \leq B$ **entonces** ALG $\leftarrow$ ALG $+ x^*$;
  $\hat{S} = \hat{S} - x^*$.
fin
devolver ALG.

(a) Sea $A = \text{ALG}$ para cualquier iteración del algoritmo y sea $x \in \hat{S}$ en ese momento.

Demuestre que $f(\text{OPT}) - f(A) \leq \frac{B}{c(x)} f_A(x)$, independiente si x es o no agregado a ALG. Aplique esta desigualdad iterativamente para obtener que $f(A+x) \geq \left(1 - \exp\left(\frac{-c(A+x)}{B}\right)\right) f(\text{OPT})$ y concluya, usando monotonía que si ALG es el conjunto final devuelto por el algoritmo, entonces para todo $x \notin \text{ALG}$, $f(\text{ALG} + x) \geq (1 - 1/e) f(\text{OPT})$.

Lamentablemente el algoritmo anterior por si solo no logra una aproximación constante partiendo de $Q = \emptyset$ ni siquiera para el caso de f aditiva (piense en un ejemplo con 2 elementos con $f(e_1) = 1, f(e_2) = \epsilon, c(e_1) = B, c(e_2) = B\epsilon/2$, la mejor solución elige e_1 con un valor de 1, sin embargo greedy decide elegir e_2 por un valor de ϵ y no puede continuar luego de eso). Sin embargo podemos usar el lema anterior para probar que un algoritmo ligeramente distinto obtiene una buena aproximación.

(b) Pruebe que el siguiente algoritmo greedy modificado devuelve un conjunto factible tal que $f(\text{ALG}) \geq \frac{1}{2}(1 - \frac{1}{e}) f(\text{OPT})$.

Algoritmo 8: Greedy modificado.

$e^* \leftarrow \arg\max_{e \in S\colon c(e) \leq B} f(e)$.
$X^* \leftarrow$ la respuesta de Greedy para knapsack. **devolver** *el mejor entre* e^* *y* X^*.

(c) Considere finalmente el siguiente algoritmo.

Algoritmo 9: Greedy desde conjuntos de tamaño 3.

1 Para todo $Q \subseteq S$, con $|Q| \leq 3$, $c(Q) \leq B$, correr algoritmo greedy para la función $f_Q \colon 2^{S \setminus Q} \to \mathbb{R}$ con una mochila de tamaño $B - c(Q)$ para obtener un conjunto ALG_Q.
2 **devolver** *el conjunto* $R = Q \cup \mathrm{ALG}_Q$ *con mayor valor* $f(R)$.

Este algoritmo es polinomial. Suponga que $|\mathrm{OPT}| > 3$ (el caso complementario es simple) y llame $e_1 = \arg\max_{e \in \mathrm{OPT}} f(x)$, $e_2 = \arg\max_{e \in \mathrm{OPT} - e_1} f_{\{e_1\}}(x)$ y $e_3 = \arg\max_{e \in \mathrm{OPT} - e_1 - e_2} f_{\{e_1, e_2\}(x)}$.

Considere el conjunto $Q = \{e_1, e_2, e_3\}$ y llame $R = Q \cup \mathrm{ALG}_Q$.

Demuestre que para todo $x \in S$, $3f_R(x) \leq f(Q)$. Aplique la parte (1) para la función SMN f_Q y un x apropiado y concluya que el algoritmo es una $(1 - 1/e)$-aproximación.

5. Programación dinámica y redondeo de datos para PTAS

Ejercicio 5.1. (2 punto) Multiple knapsack versión fraccional.

Dada una colección de m mochilas de capacidad $B > 0$ y una colección de n items, donde $v_i \geq 0$ y $s_i \in [0, B]$ son el valor y tamaño de cada item, consideramos 2 versiones del problema MK fraccional.

(a) (Versión divisible)

Cada item i debe elegir una fracción $x_{ij} \in [0, 1]$ a ser empaquetada en la mochila j. De modo tal que en total no más de 1 unidad del item i sea usado. En términos de un PL, este problema es

$$\begin{aligned} &\max \sum_{i=1}^{n} v_i \sum_{j=1}^{m} x_{ij} \\ &\sum_{j=1}^{m} x_{ij} \leq 1, \quad \forall i \in [n] \\ &\sum_{i=1}^{m} s_i x_{ij} \leq B, \quad \forall j \in [m] \\ &0 \leq x_{ij} \leq 1, \quad \forall i \in [n], j \in [m] \end{aligned}$$

Describa un algoritmo para resolver el problema anterior basándose en una solución de knapsack fraccional en una sola mochila de tamaño mB. Use esto para probar que existe una solución óptima donde cada item i o bien no es usado ($\sum_j x_{ij} = 0$), o bien es usado por completo ($\sum_j x_{ij} = 1$) pero solo es repartido en 2 mochilas, o bien es usado fraccionalmente ($\sum_j x_{ij} \in (0, 1)$) pero solo uno de los x_{ij} no es 0. Más aún, muestre que solo puede haber un objeto usado fraccionalmente.

(b) (Versión indivisible)

En este caso, cada objeto i debe elegir a lo más una mochila, y en dicha mochila puede ser asignado de manera completa o fraccional (es decir, $x_{ij} > 0$ para a lo más 1 mochila j).

Diseñe una 2-aproximación para este problema.
Indicación: use que en la solución óptima de la versión divisible cada objeto es asignado a lo más 2 mochilas.

Ejercicio 5.2. (2 puntos) Multiple knapsack fraccional indivisible. Revisitamos el problema del ejercicio anterior. Sin pérdida de generalidad, suponga que todas las densidades v_i/s_i de los items son distintas. Sea x una solución óptima, y sea $y_i = \sum_{j\in[m]} x_{ij}$. Llame a i asignado si $y_i > 0$.

(a) Demuestre que si i es asignado e i' no es asignado por este óptimo, entonces $v_i/s_i > v_{i'}/s_{i'}$ y concluya que hay un óptimo con la siguiente propiedad, existe un objeto i tal que los objetos asignados son exactamente los objetos con densidad mayor o igual que v_i/s_i. Más aún, dentro de los objetos asignados, a lo más m son asignados de manera fraccional (i.e. son i tales que $y_i \in (0, 1)$.)

(b) Encuentre un FPTAS para este problema en el caso m constante. **Indicación**: Suponga primero que conoce los objetos asignados por el óptimo y el valor de todos los $y_i s_i$ (aproximadamente). Use el FPTAS para makespan para decidir si los objetos y tamaños elegidos entran en m mochilas de capacidad B (con tolerancia $1 + \varepsilon$). Luego concluya.

Ejercicio 5.3. (2 puntos) Usando la notación del ejercicio 5.1, consideremos la versión integral de **múltiple knapsack**, es decir cada item debe o bien ser ignorado, o bien debe ser incorporado por completo en una mochila. El problema **partición** consiste en decidir, dados $2n$ números $a_1, \ldots, a_{2n} \in \mathbb{N}$ si existen una partición de $[n]$ en dos conjuntos S_1 y S_2 tal que $\sum_{i\in S_1} a_i = \sum_{i\in S_2} a_i$. Este problema es NP-completo.

(a) Demuestre que no existe FPTAS para múltiple knapsack (integal) salvo que P=NP. **Indicación:** Muestre que la existencia de tal FPTAS permite resolver de manera exacta partición.

(b) Basado en el PTAS que adivina items valiosos para knapsack, encuentre un PTAS para el caso en el que m es constante.

Ejercicio 5.4. (3 puntos) Algoritmo pseudopolinomial para (multiple) knapsack con valores arbitrarios pero tamaños enteros acotados.

Usando la notación del ejercicio 5.1, consideremos la versión integral de múltiple knapsack, es decir cada item debe o bien ser ignorado, o bien debe ser incorporado por completo en una mochila.

(a) (Una mochila) En clases vimos un algoritmo pseudopolinomial para knapsack que corre a tiempo $O(n^2 v_{\text{máx}})$. Diseñe un algoritmo alternativo para knapsack que corra a tiempo $O(nB)$. **Indicación:** Llene una tabla dinámica que para todo $j \in [n]$ y todo $b \leq B$, determine el mejor valor obtenible usando los objetos de $[j]$ en una mochila de tamaño b. No necesita explicar como se obtiene la asignación de objetos a mochilas, a partir de su valor en la tabla.

(b) (Varias mochilas) Extienda el algoritmo anterior a m mochilas, obteniendo un algoritmo para múltiple knapsack a tiempo polinomial en n y B pero tal vez exponencial en m. Este algoritmo es pseudopolinomial para el caso m constante. **Indicación:** Use una tabla $m + 1$ dimensional.

(c) Usando que las m mochilas son indistinguibles, obtenga un algoritmo que corra a tiempo polinomial en n, m y a lo más exponencial en B.

(d) (opcional) Transforme el algoritmo anterior en un PTAS

Ejercicio 5.5. (3 puntos) Caminos de largo mínimo con restricción de tiempo. Sea $G = ([n], E)$ un grafo dirigido acíclico cuyos vértices están ordenados en orden topológico (es decir, $(i, j) \in E \implies i < j$). Sean $c, \ell \colon E \to \mathbb{N}$ dos funciones. Interpretamos $\ell(e)$ como el largo del arco e y $c(e)$ como su costo. Además tenemos un valor $L > 0$.

1. Diseñe un algoritmo pseudopolinomial de complejidad $O((n+m)L)$ para encontrar un 1-n camino de costo $c(P)$ mínimo, de entre los caminos de largo total $\ell(P) \leq L$.

2. Diseñe un algoritmo pseudopolinomial de complejidad $O((n+m)nc_{\text{máx}})$ para encontrar un 1-n camino de costo $c(P)$ mínimo, de entre los caminos de largo total $\ell(P) \leq L$.

3. FPTAS mediante redondeo de costos. Suponga que tiene un método a tiempo $\text{poly}(n, m, 1/\varepsilon)$ que, dado un valor K, o bien demuestra que OPT $> K$, o bien entrega un camino de largo $\leq L$ y costo a lo más $K(1+\varepsilon)$. Use búsqueda binaria apropiadamente para obtener un FPTAS a partir de este método. Finalmente, implemente el método, redondeando apropiadamente el costo de cada arco al múltiplo entero más cercano de $K\varepsilon/(n-1)$.

6. Búsqueda local

Ejercicio 6.1 (2 puntos)**.** En el problema de MAX DICUT, la entrada es un digrafo $G = (V, E)$ y el objetivo es encontrar un conjunto $S \subseteq V$ que maximice el número de arcos que salen de S, es decir $|\delta^+(S)|$. En la versión con pesos, se da además una función $w\colon S \to \mathbb{R}_+$ y se busca encontrar S que maximice $w(\delta^+(S))$.

1. Muestre un algoritmo basado en búsqueda local que obtenga una 4 aproximación para MAX DICUT (respectivamente $4+\varepsilon$ para la versión con pesos).

2. (punto extra) ¿Puede lograr una 3 aproximación?

Ejercicio 6.2. (2 puntos) En el problema MAX EXACT k-SAT, la entrada es una familia de cláusulas $(C_i)_{i\in[m]}$ donde cada cláusula es la disjunción de exactamente k literales, y tiene un peso $w_i \geq 0$.

1. Para cada asignación σ, considere la vecindad $N(\sigma)$ dada por todas las asignaciones obtenidas al cambiar el valor de una sola asignación. Demuestre que un óptimo local para esta vecindad es una 3/2-aproximación para MAX EXACT 2-SAT. (indicación: Defina S_0, S_1, S_2 donde S_i es el conjunto de las cláusulas para las que i terminales de σ son verdaderos, y pruebe que $w(S_1) + w(S_2) \geq 2/3(w(S_0) + w(S_1) + w(S_2)))$

2. En vez de que un algoritmo use como esquema la función natural $w(S_1)+w(S_2)$ una alternativa es tratar de maximizar una función diferente. A esto se le llama *búsqueda local miope*. Suponga que busca con la misma vecindad ahora un óptimo local σ para la función $3w(S_1)+4w(S_2)$. Defina para todo j, $P_{i,j}$ como las cláusulas de S_i que tienen a x_j y $N_{i,j}$ como las cláusulas de S_i que tienen a $\bar{x}_j$ como literal. Demuestre que $N_{1,j} + 3N_{0,j} \leq P_{2,j} + 3P_{1,j}$ y use esto para concluir (sumando sobre j) que $3w(S_0) \leq w(S_1) + w(S_2)$. Deduzca que cualquier óptimo local miope es una 4/3-aproximación.

Ejercicio 6.3. (2 puntos) Considere el problema de encontrar un matching perfecto de peso máximo en un grafo completo $G = (V, E)$ de $2n$ vértices, con pesos $w\colon E \to \mathbb{R}_+$. Si bien exisen algoritmos polinomiales para este problema, buscamos en este ejercicio analizar un algoritmo de búsqueda local simple para este problema.

Sea $\ell \in \mathbb{N}$ y considere el algoritmo de búsqueda local siguiente:

Algoritmo 10: Ciclos cortos.

```
1 ALG ← Matching Greedy
2 Repetir:
3 si existe un ciclo ALG-alternante C de largo ≤ 2ℓ, con w(C \ ALG) − w(C ∩ ALG) ≥ (ε/n)w(M).
    entonces ALG ← ALGΔM;
4 en otro caso Devolver ALG;
```

Demuestre primero que el algoritmo anterior termina en a lo más n/ε iteraciones y que cada iteración se puede hacer en tiempo $O(n^{2\ell})$.

Para analizar el factor de aproximación, llame ALG al resultado del algoritmo y OPT a un matching óptimo. Considere la colección $\mathcal{C}$ de ciclos ALG-OPT alternantes en ALGΔOPT.

Sea $C \in \mathcal{C}$ un ciclo de largo $\ell(C) > 2\ell + 2$ y orientelo en una de las dos direcciones posibles. Para cada $e \in C \cap \text{ALG}$ llame X_e al ciclo obtenido de tomar el camino P de largo $2\ell - 1$ en C, partiendo de la arista e y cerrándola con una cuerda fuera del ciclo. Sumando X_e sobre todos los aristas $e \in C \cap \text{ALG}$ deduzca que

$$w(\text{ALG} \cap C)(\varepsilon/n)|\text{ALG} \cap C| > (\ell - 1)w(\text{OPT} \cap C) - \ell w(\text{ALG} \cap C)$$

Haga algo similar para los ciclos cortos y deduzca, sumando sobre todo $C \in \mathcal{C}$ que el algoritmo es una $(\ell + 1 + \varepsilon)/\ell$-aproximación.

7. Primal-Dual

Ejercicio 7.1. (3 puntos) Primal-dual para multicut en árboles. Dados $T = (V, E)$, un árbol, con costos $(c_e)_{e \in E} \geq 0$ en sus aristas y una colección $(s_i, t_i)_{i \in [k]}$ de pares de terminales, considere el problema de encontrar una colección de aristas $F \subseteq E$ tal que al removerlos todos los pares s_i-t_i son separados. Para cada $u, v \in V$, llamamos $P_{u,v}$ al único u-v camino.

1. (1 punto) Describa una formulación como PLE del problema multicut descrito. Escriba su relajación lineal y su dual (como los costos son no-negativos, no necesita imponer $x_e \leq 1$ en el PL).

2. (2 puntos) Tome un vértice r arbitrario como raíz. Para cada par de vértices u, v llame $d(u, v)$ al número de aristas de $P_{u,v}$. Llame además $(u \wedge v)$ al vértice x de $V(P_{u,v})$ que minimiza $d(x, r)$.

   Considere el algoritmo primal-dual estándar usando la formulación anterior, pero imponiendo que en cada etapa se aumenta la variable asociada a la restricción no satisfecha indexada por el índice i tal que $d(s_i \wedge t_i, r)$ es mínimo. Muestre que este algoritmo es una 2-aproximación.

8. Redondeo aleatorio

Ejercicio 8.1. (3 puntos) Considere la siguiente relajación del problema de conjunto estable más grande en un grafo $G = (V, E)$

$$\begin{aligned} \text{máx} \sum_{v \in V} x_v \\ x_u + x_v \leq 1, \quad \forall uv \in E \\ 0 \leq x_v \leq 1, \quad \forall v \in V. \end{aligned}$$

Llame x^* a una solución óptima del PL, m al número de aristas y considere el siguiente algoritmo.

(1) Si $\sum_{v \in V} x^* < 2\sqrt{m}$, elegir un vértice v_0 cualquiera y devolver ALG $= \{v_0\}$. En caso opuesto continuar.

(2) Construya un conjunto aleatorio $S \subseteq V$, poniendo cada $v \in V$ en el conjunto S de manera independiente con probabilidad $p_v = x^*_v/\sqrt{m}$.

(3) Llame F al conjunto de aristas con ambos extremos en S, y sea $V(F)$ todos los vértices cubiertos por F. Devuelva ALG $= S \setminus F$.

El objetivo del problema es demostrar que el algoritmo anterior es una $2\sqrt{m}$ aproximación

1. (1 punto) Demuestre que el algoritmo devuelve un conjunto estable, y que si $\sum_{v \in V} x^* < 2\sqrt{m}$ entonces ALG es una $2\sqrt{m}$-aproximación.

2. (2 puntos) Suponga entonces que $\sum_{v \in V} x^* \geq 2\sqrt{m}$. Llame Y_v a la indicatriz del evento $v \in S$ y Z_e a la indicatriz del evento $e \in F$. Calcule una cota inferior para $\mathbb{E}[\text{ALG}]$ en función de las variables X_u y Z_e. Pruebe, usando desigualdades conocidas, que $\Pr(Z_e = 1) \leq \frac{1}{2m}$ y use esto para concluir que $\mathbb{E}[\text{ALG}]$ es una $2\sqrt{m}$-aproximación en este caso también.

9. Programación Semidefinida Positiva

Ejercicio 9.1. (3 puntos) Podemos usar SDP para dar mejores algoritmos para MAX SAT. En esta tarea solo veremos el caso MAX 2SAT en el cual cada clausula tiene a lo más 2 literales.

1. (1 punto) Siguiendo la idea de la formulación para MAX CUT, exprese MAX 2SAT como un problema de optimización (cuadrático) exacto con variables $y_i \in \{-1, 1\}$ para cada variable x_i de la fórmula booleana dada. Su formulación debe ser *lineal* en los términos $(y_i y_j)_{i,j}$. *Indicación:* Cree una variable adicional $y_0 \in \{-1, 1\}$ que ayude a indicar cual valor (-1) o 1 es *verdadero*.

2. (2 puntos). Escriba una relajación del programa anterior como un programa vectorial (SDP) y use las mismas ideas que en MAX CUT para obtener una 0.878 aproximación para MAX 2SAT.

10. Redondeo Iterativo

Ejercicio 10.1. (3 puntos) Sea $\{I_1, \ldots, I_n\}$, una familia de n intervalos en el plano con $I_i = [a_i, b_i]$ para $i \in [n]$. Considere además *pesos* (arbitrarios) asociados $w_1, \ldots, w_n \geq 0$. Sea k un entero positivo ($k \geq 1$). Decimos que $J \subseteq [n]$ es un k-empaquetamiento si cada punto $p \in \mathbb{R}$ pertenece a lo más k intervalos de $\{I_j : j \in J\}$. Si J es un k-empaquetamiento, llamamos $w(J) = \sum_{j \in J} w_j$ a su peso.

Sin pérdida de generalidad podemos suponer que todos los $\{a_1, a_2, \ldots, a_n, b_1, \ldots, b_n\}$ son distintos.

1. (1.5 puntos) Escriba un PL que sea una relajación natural para el problema de encontrar un k-empaquetamiento de peso máximo. Muestre primero que basta usar a lo más $2n$ restricciones que no sean del tipo $0 \leq x_i \leq 1$. Luego demuestre que en realidad solo necesita imponer $n - 1$ restricciones. Para esto último, le puede ser útil notar que en muchos casos la restricción para un cierto $p \in \mathbb{R}$ es equivalente a la restricción para $p + \varepsilon$. Una vez que llegue a n restricciones, pregúntese si hay alguna que es innecesaria.

2. (1.5 puntos) Use redondeo iterativo para mostrar que el PL encontrado anteriormente es integral.