Trabajo Práctico
75.29 Teoría de Algoritmos I

FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Profesores:
Rosita Wachenchauzer

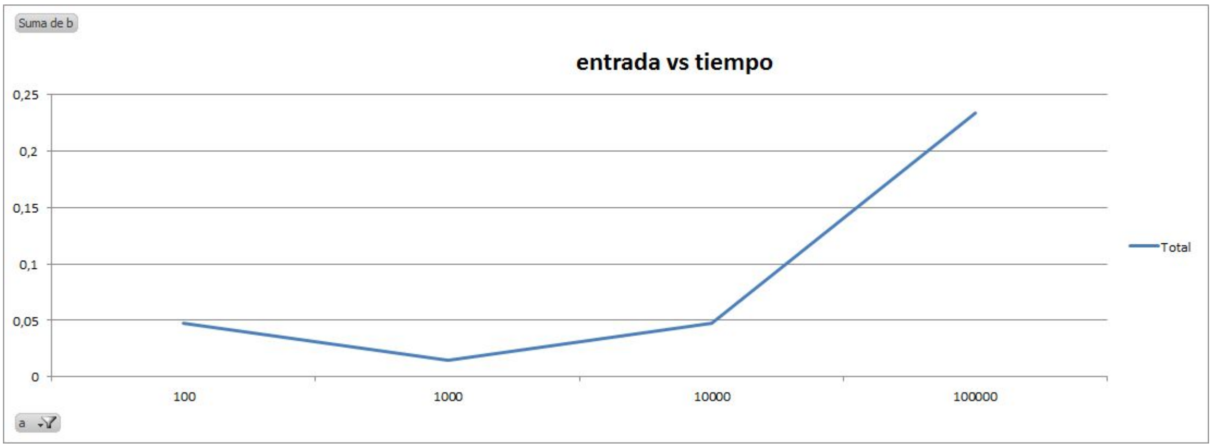
1º Cuatrimestre 2017

Integrantes:

NOMBRE Y APELLIDO	PADRÓN
Hernán Arroyo García	91257
Damian Cassinotti	96618
Mariano Cinalli	95456

Programación Dinámica

El algoritmo sellAction creado lo que muestra es el día de compra y día de venta de acciones de tal manera que la ganancia sea máxima en un lapso de N días.
El algoritmo es Botton-Up, va tomando los datos desde el inicial al final, Inicializa los parametros dateToBuyActual y dateToSell (cuando se tienen dos día), después va recorriendo y por cada dia, realiza una serie de condiciones para modificar las variables dateToBuyActual y dateToSell ; si se tiene un dia donde el valor de la acción es mas grande que el anterior se modifica dateToSell , si el valor de la acción es menor que dateToBuyActual , se guarda este dia en la variable dateToBuyFollowing, porque puede que haya otra mejor solución, si existe una mejor solución se actualizan las variables dateToBuyActual y dateToSell.
La complejidad del algoritmo es 0(n).



Algoritmos randomizados

El algoritmo de Karger se utiliza para obtener un corte de un grafo conexo no dirigido, con tendencia a encontrar el corte mínimo.
Este algoritmo se basa en la contracción de aristas. La arista (*u, v*) a contraer se elige al azar de todo el set de aristas del grafo. La contracción se basa en concatenar los nodos u y v, convirtiéndolos en un nuevo nodo uv. Luego, todas las aristas (*w, u) y (w, v)* se convierten en (w, uv), con w distinto de u y de v. Además, se eliminan todas las aristas del tipo (u, v). El proceso de contracción se realiza hasta que queden solamente 2 nodos en el grafo. Entonces, el corte mínimo será el único corte posible de nuestro grafo resultante.
En nuestra implementación cada contracción de aristas se realiza en O(|E|), ya que debemos recorrer todas las aristas del grafo de forma tal de actualizar las referencias a los nodos. Como se realizan |V| - 2 contracciones (pues en cada una se elimina un nodo, y se continúa hasta que queden 2), el orden temporal del algoritmo será O(|V|·|E|).
Mediante las distintas pruebas realizadas, llegamos a la conclusión de que con $|V|^2$ iteraciones obtuvimos el corte mínimo en la mayoría de los casos.
De esta forma, el algoritmo termina siendo $O(|V|^4)$, pues en nuestros grafos se cumple la condición $|E| = 2|V|$.

Este algoritmo de randomización corresponde a la categoría Monte Carlo, pues es siempre rápido (es polinomial), y tiene una alta probabilidad de devolver el resultado correcto.
A continuación se muestra un gráfico de los tiempos de ejecución del algoritmo en correlación con la cantidad de aristas del grafo:

Algoritmos aproximados

Este algoritmo es una aproximación al problema de subset sum.
inputs:
Una lista de valores (mayores a cero), un número máximo t y un parámetro e que llamamos parámetro de aproximación.
output:
Valor aproximado a t formado por la suma de valores dentro del array.

El algoritmo une una sublista de la ingresada por parámetro, con la misma sublista más el valor en la posición i de la lista original.
Mantenemos en listas auxiliares valores representativos de la lista original.
Los valores representantes son elegidos por la función trim_list, que remueve los valores cercanos al último que le es agregado.
Como la lista que es parámetro de trim_list está ordenada y vamos agregando valores, a una nueva lista, comenzando desde el primero, e ignorando los cercanos al último, nos aseguramos que los valores de la lista son cercanos a los originales y que la cantidad de valores se ve reducida.
Removemos los valores mayores a t.
Luego de n iteraciones, devolvemos el mayor valor de la última lista.

La clave para mantener un tiempo de ejecución polinomial está en mantener el tamaño de las listas en tamaño polinomial.
Al lograr esto todas las operaciones sobre las mismas serán polinomiales.