Teoría de Algoritmos I

Primer Cuatrimestre 2017

Trabajo Práctico 3

Integrante	Padrón	Correo electrónico
Rodrigo De Rosa	97799	rodrigoderosa@outlook.com
Marcos Schapira	97934	schapiramarcos@gmail.com
Facundo Guerrero	97981	facundoiguerrero@gmail.com

Índice

1. Programación Dinámica

En esta sección se analiza una solución al problema de la predcción de acciones a través de la programación dinámica.

1.1. Algoritmo

El algoritmo utilizado para resolver el problema planteado esta basado en el algoritmo de kadane. Este busca la maxima suma de elementos contiguos dentro de un arreglo.

1.1.1. Funcionamiento

Este algoritmo funciona de la siguiente forma:

- Inicializa un `día de compra`, un `día de venta`, un `día de compra auxiliar`, todos como el primer dia. Tambien se inicializa una `ganancia máxima` y una `ganancia temporal`, ambas como 0 ya que, hasta el momento, el dia de compra es igual al dia de venta.
- Luego itera sobre todos los días (valores diferentes de acciones) verificando si en el día actual(dia i) es más o menos favorable comprar acciones que en el día en el que se pretendía hacerlo hasta el momento(dia k), determinando el `día de compra auxiliar`. Esto asegura la obtencion de la mayor ganancia hasta el dia i-1.
- A partir del día que determinó, calcula la `ganancia temporal` como la ganancia que se obtendría si las acciones fueran compradas en el `día de compra` y vendidas el `día actual`. Luego se verifica si la `ganancia temporal` es mayor a la `ganancia máxima`.
- En tal caso, determina el `día de venta` como el actual, el `día de compra` como el que previamente era el `día de compra auxiliar` y la `ganancia máxima` como la que era la `ganancia temporal`.
- Al finalizar la iteración, queda determinado el `día de compra` más conveniente, el `día de venta` más conveniente y la `ganancia máxima` obtenible.
- Dado que el algoritmo propuesto recorre una sola vez el arreglo, funciona en $O(n)$.

1.1.2. Ecuación de recurrencia

Para la ecuación de recurrencia se plantea lo siguiente:

- Se tiene una variable $\mathtt{C}_i$`= Día en el que se compran las acciones hasta el paso i,con i=1,...,n`.
- Ademas, se tiene otra variable $\mathtt{V}_k$`= Día en el que se venden las acciones hasta el paso k, con k=i,...,n`.
- Observar que k esta relacionada con i, ya que el dia de venta debe ser mayor o igual al dia de compra. En el caso de ser igual, la ganancia seria 0.
- Se puede obtener la `Ganancia Temporal` del paso `i,k`, como $\mathtt{GT}_{i,k} = V_k - C_i$.
- Entonces, la `Ganancia Máxima` es la máxima $\mathtt{GT}_{i,k}$.
- Se puede definir la `Ganancia Máxima` para el paso `i,k` como:

$$\mathtt{GM}_{i,k} = \begin{cases} V_k - C_i & \mathtt{si}\ GT_{i,k} > GM_{i,k} \\ GT_{i,k-1} & \mathtt{si}\ GT_{i,k} \leqslant GM_{i,k} \end{cases}$$

2. Algoritmos Randomizados

En esta sección se analiza una solución al problema de hallar el corte global mínimo en un grafo no dirigido a través de un algoritmo randomizado.

2.1. Algoritmo

Para resolver este problema se utilizó el algoritmo de Karger descripto en la bibliografía proporcionada por la cátedra.

2.1.1. Funcionamiento

Sea el grafo $G = (E, V)$, el procedimiento del algoritmo es el siguiente:

- Mientras $|V| > 2$:
    - Se elige $e(u, v) \in E$ aleatoriamente.
    - Se crea un $w \in V$, el cual reemplaza tanto a u como a v en todas las aristas en las que se encuentran. Es decir, w puede tener más de una arista que vaya a un mismo vértice $q \in V$.
    - Se elimina $e(u, v)$ de E.
    - Si existe alguna $e(v, v) \in E$ (arista de un vértice consigo mismo), se elimina.
- Se devuelven las aristas que unen a esos dos vértices como el corte mínimo.

2.1.2. Categoría de randomización

Es un algoritmo *Monte-Carlo* porque para algún orden de selección aleatoria de aristas, el corte obtenido *no* es el mínimo. Es decir, es rápido siempre pero no siempre da resultados correctos.

La probabilidad de que este algoritmo devuelva un corte que sea mínimo es $p \geqslant \binom{n}{2}^{-1}$ con $n = |V|$. Un dato adicional es que si el algoritmo se corre $T = \binom{n}{2} \ln n$ veces, la probabilidad de no encontrar un corte mínimo es $[1 - p]^T \leqslant \dfrac{1}{n}$ en un tiempo $O(Tm) = O(n^2 m \log n)$ con $m = |E|$.

3. Algoritmos Aproximados

En esta sección se analiza una solución al problema de la suma de subconjuntos a través de un algoritmo aproximado. Para resolver este problema se utilizó la estrategia polinómica descripta en la bibliografía proporcionada por la cátedra.

3.0.1. Funcionamiento

El problema de la suma de subconjuntos (subset sum) consiste en, a partir de un conjunto S de enteros positivos y un target t también entero positivo, saber si existe algún subconjunto de S cuya suma sea exactamente t. Este problema es NP-Completo.

A partir de él se puede derivar a una aproximación completamente polinómica mediante el "recorte" o "trimming" de cada subconjunto que se va generando en el algoritmo exacto. Este mecanismo se sostiene de la idea de que si dos números pertenecen a S y tienen valores similares entonces no tiene mucho sentido mantener a ambos explícitamente (en referencia al algoritmo aproximado). Así es como mediante un parámetro de aproximación sigma tal que: $0 < \sigma < 1$

Se eliminan tantos elementos de S como sea posible ya que por cada elemento eliminado va a haber otro que pertenezca a S y lo represente. Así es como el algoritmo logra dado un conjunto S y un parámetro t devolver la mayor suma de elementos menor o igual a t. A la vez el algoritmo obtiene por parámetro a sigma, con lo cual la suma que devuelve está a un factor de $(1 + \sigma)$ del valor real.

3.0.2. Análisis del Algoritmo

La tabla a continuación muestra resultados del algoritmo con sigmas variables. A la vez los elementos en cada instancia y el valor de t fueron generados aleatoriamente. Z es el valor devuelto por el algoritmo.

Cuadro 1: Resultados para N = 350

T	Sigma	Zreal	Z real porcentual
2213	0,94	2211	%99,9
2246	0,46	2245	%99,9
2182	0,65	2182	%100
2620	0,74	2618	%99,9
173	0,62	172	%99,4

Como se puede apreciar, los porcentajes son extremadamente altos y caen dentro del factor esperado.

4. Ejecución de programas

Para correr cada algoritmo, se debe ejecutar el archivo principal de cada uno. Esto se hace de la siguiente forma:

En la carpeta `Programación Dinámica` abrir la consola y ejecutar `python main.py`
En la carpeta `Algoritmos Randomizados` abrir la consola y ejecutra `python main.py`
En la carpeta `Algoritmos Aproximados` abrir la consola y ejecutra `python main.py`