Problema B - A Pyramid Scheme

Nome: Fábio Miguel Rodrigues Vaqueiro **Nº estudante:** 2019222451

Nome: Marta Carreira Santos **Nº estudante:** 2019220054

Nome Mooshak: 2019222451_2019220054

1. Descrição do Algoritmo

O algoritmo é baseado num algoritmo top-down (de programação dinâmica), onde se começa por analisar todo um conjunto de nós, árvore correspondente a todo o esquema de pirâmide, que posteriormente será dividindo em partes.

Deste modo, começa-se por analisar a raiz da árvore, de seguida, avança-se na árvore dividindo esta em pequenos segmentos, até encontrarmos o último elemento, isto é, uma folha da árvore. Uma vez que, o algoritmo é recursivo, feitos os cálculos para o último elemento desse segmento recua-se ao nó pai e assim sucessivamente, voltando ao topo da árvore.

Inicialmente, foi implementada uma função recursiva para percorrer toda a árvore de nós. Nesta, recebe-se como entrada um id, de modo a ser possível procurá-lo no conjunto de nós, ou seja, através do id é feita uma procura no *map*, não sendo necessário ter um ciclo para percorrer todos os nós da árvore.

Posteriormente, percorrem-se todos os filhos do próprio nó, ou seja, os nós recrutados por este no esquema de pirâmide, sendo atualizadas as variáveis de quando o nó não é investigado guardando-se a soma dos valores dos filhos quando estes são investigados. Caso o nó seja investigado guarda-se o melhor valor de cada filho (entre o valor de ser investigado ou não).

São ainda feitas verificações de modo a obter-se o mínimo de nós a investigar com o máximo de valor dos pagamentos somados, na atualização dessas variáveis.

2. Estruturas de dados

Para as estruturas de dados foram implementadas uma classe *No* e um hashmap *mapa,* que representa toda a pirâmide.

A classe, de modo a guardar as informações sobre cada nó, é constituída por variáveis com os valores do id, os nós recrutados por este nó, o pagamento de quando este nó foi recrutado e são guardados em dois arrays os valores mínimos do número de nós a investigar com o máximo de pagamento desses nós até ao nó, caso esse seja investigado ou não.

O mapa interliga o valor do id de um nó com o seu objeto, de modo a melhorar a facilidade de procura de um nó através do seu id, sendo assim não necessário percorrer todos os nós até o encontrarmos.

3. Correções

Tendo em conta o algoritmo implementado e a cotação máxima alcançada no mooshak, a abordagem sobre o problema feita foi a correta. Para se obter este resultado positivo foi essencial a implementação de um hashmap, desta forma é possível ir buscar o nó que se pretende através do id, ao invés de percorrer todos os nós. Portanto, não é necessário ter uma lista de nós ordenada.

Para facilitar no acesso às variáveis de cada nó na árvore inserido foi implementada uma classe com as variáveis *id*, *custo*, dois arrays *usado* e *n_usado* e um vector *recrutados*.

4. Análise do Algoritmo

Na função principal do algoritmo, existe um ciclo que percorre os filhos de cada nó da árvore, e para cada filho uma chamada recursiva da própria função. Deste modo, esse ciclo, no pior caso, itera n vezes – complexidade O(n).

Assim sendo, a complexidade temporal e a complexidade espacial para este algoritmo será: O(n).

5. Referências

- Material disponibilizado no UCStudent

- Algoritmo DFS

- Programação dinâmica

- Documentação c++

- Documentação sobre Hashmap

- https://iq.opengenus.org/vertex-cover-problem/

- https://www.geeksforgeeks.org/vertex-cover-problem-set-2-dynamic-programming-solution-tree/