Relatório de Programação para o Problema 2

Equipa:
ID de estudante: 2019219581 Nome: Tatiana Silva Almeida
ID de estudante: 2019220082 Nome: Sofia Santos Neves

1. Descrição do algoritmo

De uma forma geral, o que o algoritmo faz é: Percorre todos os nós, usando recursão, até chegar às folhas. Após chegar a uma folha (caso base), calcula, para a folha em questão, os valores de marcar o nó e de não marcar o nó: (1, Preço de entrada desta pessoa) e (0, 0), respectivamente, colocando esses dois valores nesse mesmo nó.

Após o calculo dos valores ‘’*used_money*’’, ‘’*used_node*’’, ‘’*unused_money*’’ e ‘’*unused_node*’’ para cada folha, cada nó pai irá calcular também os valores de marcar-se a si ou de não se marcar.

Para o caso em que o nó não está marcado:

- ➔ A variável ‘’*unused_node*’’ do nó, passa a ter o valor da soma dos valores de *‘’used_node”* dos filhos do nó em questão. Isto deve-se ao facto de que, sempre que o próprio nó não se encontra marcado, é obrigatório que os seus filhos estejam marcados de modo a que não se perca a ligação entre pai-filho.
- ➔ A variável “*unused_money*’’ do nó, pela mesma justificação anterior, tem que conter a soma dos ‘’*used_money*’’ dos filhos do nó em questão.

Para o caso em que o nó está marcado:

- ➔ A variável ‘’*used_node*’’ do nó, passa a ter o valor de 1 mais a soma dos melhores valores dos filhos do nó em questão. Isto deve-se ao facto de que, sempre que o próprio nó se encontra marcado, os seus filhos podem ou não estar marcados. Neste caso, é importante escolher a situação mais benéfica.
- ➔ A variável “*used_money*’’ do nó, pela mesma justificação anterior, tem que conter a soma entre o valor pago pela pessoa em questão e os melhor valores dos seus filhos.

  Para escolher o melhor valor dos filhos, existem três possibilidades:

  1. **Se o número de nós com o meu filho marcado for inferior ao número de nós com o meu filho não marcado** então uso os valores do meu filho como estando marcado.
  2. **Se o número de nós com o meu filho não marcado for inferior ao número de nós com o meu filho marcado** então uso os valores do meu filho como não marcado.
  3. **Se o número de nós com o meu filho marcado e não marcado for igual, então** quer-se selecionar a opção em que a soma de dinheiro pago por todos os nós marcados seja maior. Para tal, há duas opções:
     a. **O valor guardado no ‘’used_money’’ do filho é superior ao valor guardado no ‘’unused_money”** então implica usar o filho marcado.

b. **O valor guardado no ‘’used_money” do filho é inferior ou igual ao valor guardado no ‘’unused_money”** então implica usar o filho não marcado.

Para a impressão do output final, escolhemos a partir do nó 0 qual é a opção que tem um menor número de nós marcados. Caso ambas tenham o mesmo número de nós, escolhemos a que corresponde a uma maior soma de dinheiro pago pelos nós marcados.

2. Estruturas de dados

Como estruturas de dados, usamos apenas um vetor de *structs*, onde cada struct corresponde a uma pessoa do esquema em pirâmide.

Para cada pessoa (nó) guardamos vários inteiros:

- ➔ ***id*** **-** corresponde ao número da pessoa dentro do esquema em pirâmide;
- ➔ ***cost*** **-** corresponde ao preço que a pessoa pagou para entrar no esquema;
- ➔ ***used_node*** **-** corresponde à soma entre o próprio (um nó) e o número de nós que estarão marcados em toda a sub-pirâmide inferior à pessoa em questão, caso esta seja marcada;
- ➔ ***used_money*** **-** corresponde ao valor pago pela pessoa para entrar no esquema mais a soma dos valores pagos pelos nós marcados na sub-pirâmide da pessoa em questão;
- ➔ ***unused_node*** **-** corresponde ao número de nós que estarão marcados, em toda a sub-pirâmide inferior à pessoa em questão, caso esta não esteja marcada;
- ➔ ***unused_money*** **-** corresponde à soma de todos os *used_money* dos filhos do nó em questão, tendo em conta que este não vai ser marcado.

3. Correção

Sub-problema

Encontrar a melhor solução entre marcar o nó X ou não marcar o nó X, com o objectivo de ter o mínimo de nós marcados e, em caso de empate, escolher o maior valor.

Estrutura ótima

1) **Assunção**
   - **a)** *used_nodes* e *used_money* de um nó i contém a melhor solução para o caso em que i é usado.
   - **b)** *unused_nodes* e *unused_money* de um nó i contém a melhor solução para o caso em que i não é usado.

2) **Negação**
   - **a)** Existe uma melhor opção do que somar 1 com o somatório entre o melhor (menor número de nós ou, em caso de número de nós igual, um maior valor de entrada) dos valores dos filhos.

b) Existe uma melhor opção que apenas escolher o somatório entre os valores de *used* dos filhos.

3) Consequência

O valor final teria um menor número de nós marcados do que a estrutura óptima, ou o mesmo número de nós marcados mas com um valor superior.

4) Contradição

A nossa solução não é ótima.

4. Análise do algoritmo

Complexidade temporal: O(n)
Complexidade espacial: O(100001)

5. Referências

[1] - https://en.cppreference.com/w/