2ª Prova de EDA-II - Curso: Engenharia de Software

Data: 25 de Outubro **Horário limite:** 9:55

(20 pontos) 1) Considere as técnicas de busca sequencial, busca binária e busca baseada em hashing:

a) Descreva as vantagens e desvantagens de cada uma dessas técnicas, indicando em que situações você usaria cada uma delas.

b) Qual é a eficiência de utilização da memória (relação entre o espaço necessário para dados e o espaço total necessário) para cada método?

(10 pontos) 2) Imagine que você tenha um *bug* em sua implementação de tabela hash utilizando hash-dupla (double-hashing) de tal forma que a primeira ou a segunda função de hash retornam sempre o mesmo valor (porém diferente de 0). Descreva o que ocorre (exemplo: os custos de inserção e busca permanecem o esperado?) quando:

a) a primeira hash está errada;

b) a segunda hash está errada;

c) ambas funções de hash estão erradas.

(80 pontos) 3) Classificados de Sementes

A famosa empresa Estudos e Desenvolvimentos Agrônomos (EDA) está de volta a pedir sua ajuda.

O problema é parecido, mas é outro. A nossa querida EDA resolveu fazer a segunda versão do sistema de classificados de sementes e ele funciona em modo *streaming*, ou seja, as sementes vão chegando ao longo da execução do programa, logo não sabemos quantas sementes devem ser processadas de antemão.

A pergunta permanece a mesma: Informe as `k` melhores sementes com as informações que temos até agora.

Entrada

A entrada é composta por diversas linhas. Cada linha da entrada possui 2 números inteiros S ($0 \leq S \leq 2^{50}$) e N ($-10^6 \leq N \leq 10^6$), representando o código da Semente e a Nota da qualidade, respectivamente. A última linha, do caso de teste, contém S valendo 0 significa que é um pedido de consulta, e N será, garantidamente um número positivo representando o pedido de k melhores sementes.

Os códigos S são únicos e as notas não.

Saída

A saída deverá conter as k melhores sementes ordenadas pelos códigos da semente.

Tarefa

A sua tarefa é dividida em várias partes e todas são importantes.

EDA quer que as sementes sejam guardadas em uma tabela HASH. As colisões devem ser tratadas em uma estrutura de árvore binária de busca (BST).

Em cada um dos itens abaixo deixe a complexidade aproximada da sua implementação, se ela tende a $O(1)$; $O(lgN)$; $O(N)$; $O(NlgN)$; $O(N^2)$ ou outro.

a) (10 pontos) O que você vai usar como chave para indexar na hash? Qual deve ser o tamanho de sua tabela hash (quantos elementos ela deve indexar [sem considerar as colisões])? Implemente a função `long Hash(long key)` que deve receber a chave e retornar a Hash para a chave.

b) (20 pontos) Implemente a função `void insertItem(HashTable *hashtable, Item item)` que deverá inserir uma semente armazenada em uma struct do tipo Item em uma tabelahash do tipo HashTable

- Mostre todas as funções auxiliares necessárias. Lembre-se que as colisões devem ser tratadas com uma árvore BST, e ela deve ser implementada aqui.

c) (10 pontos) Implemente a função `int subtreecount(node *h)` que conta a quantidade de elementos em uma subárvore e armazene na variável `h->N`.

- É possível que esta contagem faça parte da função de inserção na árvore? Se você implementar esta funcionalidade embutida na inserção da árvore, recebe **(5 pontos)** extras.

d) (15 pontos) Implemente a função `nodeT* getKfromkey(HashTable *hashtable, int k, long key)` que pega os k menores elementos com uma determinada chave *key*, você pode achar conveniente implementar uma função parecida com: `nodeT* partR(nodeT* h, int k)`, que coloca na raiz o k-ézimo menor elemento da árvore.

e) (25 pontos) Implemente a função `void printKbetter(HashTable *hashtable, int k)`, que imprime as k melhores sementes armazenadas na tabela hash ordenadas pela *nota* da semente, sementes com a mesma nota devem ser impressas ordenadas pelo código da semente.

- você **não** pode copiar as sementes para um vetor e ordená-los.

Exemplo

Entrada:

```
30553 5010
27183 26616
2414 -30329
16682 23006
20027 5010
10315 32560
23488 17242
26565 -22549
2660 23760
10568 27930
0 4
```

Saída:

```
2414 -30329
26565 -22549
20027 5010
30553 5010
```

Entrada:

```
10 232
32 656
4535 -222
56 -222
767 -222
943 -222
323 -222
0 3
```

Saída:

```
56 -222
323 -222
767 -222
```