TRABALHO DA UNIDADE 1 - Análise de Algoritmos

Disciplina: IMD0032 - EDB II
Semestre: 2014/I
Professor: Carlos A. Prolo
Alunos: Bernardo Gurgel, Thiago César

1 Questão

a) Nesta questão, as instruções mais executadas são a da linha 26, a condição da linha 25, o k++ da linha 24 e o $k < n$ da linha 24. Mesmo que o número de vezes de execução dessas linhas sejam diferentes ($k < n$ é a mais executada), assintoticamente é a mesma coisa. O número de vezes que $k < n$ é executada é:

$$\sum_{j=1}^{n-1} \sum_{k=j+1}^{n-1} 1$$

No entanto, não é necessário resolvermos exatamente estes somatórios para concluir que, tanto no melhor caso quanto no pior caso, o resultado é $\theta(n^2)$. Nesse caso, em que há dois *for* embutidos, é uma fração quadrática $\theta(n^2)$.

b) Complete usando notação assintótica: O número de comparações feitas pela algoritmo no pior caso é $\underline{\theta(n^2)}$. No entanto o número de SWAPS no pior caso é apenas $\underline{\theta(n)}$.

c)

2 Questão

a)

b)

c)

3 Questão

a)

b)

c)

d)

4 Questão

Entre as muitas aplicações do *hash*, referimos a implementação eficiente dos métodos de tabelação. Estes métodos são usados em pesquisas heurísticas, e em jogos, por exemplo, para guardar o valor de configurações de um tabuleiro de xadrez. Quando inserimos um valor x e posteriormente um valor y com $h(y) = h(x)$, temos uma *colisão*. A posição da tabela para onde y deveria ir já está ocupada e terá que existir um método para resolver as colisões.

A probabilidade p de se inserir N itens consecutivos sem colisão em uma tabela de tamanho M é:

$$p = \frac{M-1}{M} \times \frac{M-2}{M} \times ... \times \frac{M-N+1}{M} = \prod_{i=1}^{N} \frac{M-i+1}{M} = \frac{M!}{(M-N)!M^N}$$

Assim, uma das formas de resolver as *colisões* é simplesmente construir uma lista linear encadeada para cada endereço da tabela, desse modo, todas as chaves com o mesmo endereço são encadeadas em uma lista linear.

A função de inserção, nesse caso, *CHAINED-HASH-INSERT(T, x)* insere o elemento x na cabeça da lista $T[h(x.key)]$. Portanto, o tempo de execução no pior caso, para a operação de inserção, é $\theta(1)$.

5 Questão

```cpp
#include <iostream>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <limits.h>

using namespace std;


int partition ( int a[], int n ){
    int sum = 0;
    for(int i = 0; i < n; i++)
        sum += a[i];

    int *s = new int[sum+1];

    s[0] = 1;
    for(int i = 1; i < sum+1; i++)
        s[i] = 0;

    int diff = INT_MAX , ans;

    for(int i = 0; i < n; i++){
        for(int j = sum; j >= a[i]; j--){
            s[j] = s[j] | s[j-a[i]];
            if(s[j] == 1){
                if(diff > abs(sum/2 - j)){
                    diff = abs(sum/2 - j);
                    ans = j;
                }
            }
        }
    }
    return sum-ans-ans;
}

```

```
int main(){
    int n,result, arr[300];
    cin >> n;

    for(int i = 0; i < n; i++){
        cin >> arr[i];
    }

    result = partition(arr,n);

    /* Se a diferenca entre a soma dos subconjuntos for igual a 0
     * entao o conjunto pode ser dividido.
     */
    if(abs(result) == 0)
        cout << "O conjunto informado pode ser dividido em dois subconjuntos com somas iguais." << endl;
    else
        cout << "O conjunto informado NAO PODE ser dividido em dois subconjuntos com somas iguais." << endl;

    return 0;
}
```

Listing 1: Programa que exibe uma mensagem se a soma dos valores de um conjunto A é igual à soma dos valores em S-A.

Suponhamos que temos $n = 500$ números, no entanto, sabemos que a soma de todos os números é no máximo $N = 10000$. Esse pequeno detalhe faz com que o problema possa ser resolvido com tempo de execução igual a $\theta(n \times N)$.