Lista 7

Victor Sena Molero - 8941317

5 de maio de 2016

Ex 23. *Festival de Estátuas de Gelo*

Resposta. Dado um m e um vetor a de n inteiros $a_1, a_2, \ldots, a_n$, temos que encontrar um vetor $x \in \mathbb{N}^n$ que minimize $\sum_{i=1}^{n} x_i$ sujeito a $x^T a = m$. Ou seja, devemos conseguir somar os inteiros dados no vetor a de forma a atingir um valor exatamente m, podendo usar cada valor de a quantas vezes forem necessárias e minimzando a quantidade de somas feitas. Vamos chamar a solução ótima de uma instância com um m dado de $f(m)$.

Temos que $f(0) = 0$ pois não precisamos somar nenhum inteiro para atingir o valor 0. Agora, queremos descobrir o valor de $f(m)$ dados os valores de todas as instâncias menores, ou seja, sabendo os valores para todo $f(x)$ tal que $x < m$. Sabemos que, se temos um vetor de a e queremos atingir a soma m, devemos escolher algum valor de a e colocar nesta soma, porém, após inserir este valor x, temos que conseguir somar $m - x$ com os mesmos valores a do problema anterior, ou seja, este é um subproblema do problema anterior.

Podemos, então, concluir a seguinte recorrência sobre um valor de $f(m)$ quando $m > 0$

$$f(m) = \min_{i=1}^{n} f(m - a_i) + 1$$

Para isso, podemos criar uma tabela que memoriza o resultado da função para todo m e calcular os valores dela crescentemente em m, sabendo que $f(0) = 0$, ou seja, podemos aplicar programação dinâmica.

Para calcular cada um dos valores, precisamos iterar por todo o vetor a, ou seja, cada cálculo de estado custa $O(n)$. Já que temos m estados para calcular, o tempo total de execução do programa será $O(nm)$.

Eu tive minha submissão aceita no URI, meu user lá é `Victor Sena Molero`. Segue o código submetido no juíz, em C++. □

```cpp
#include <bits/stdc++.h>

using namespace std;
typedef unsigned long long int ull;
typedef long long int ll;

#ifndef ONLINE_JUDGE
#define DEBUG(...) {fprintf(stderr, __VA_ARGS__);}
```

```
#else
#define DEBUG(...) {}
#endif

const int N = 30;
const int K = 1123456;

int memo[K];
int turn;
int n, m;
int v[N];
int t;

int main () {
    scanf("%d", &t);
    while (turn++ < t) {
        scanf("%d %d", &n, &m);
        for (int i = 0; i < n; i++)
            scanf("%d", &v[i]);

        memo[0] = 0;
        for (int i = 1; i <= m; i++) {
            memo[i] = K;
            for (int j = 0; j < n; j++)
                if (v[j] <= i)
                    memo[i] = min(memo[i], memo[i-v[j]]+1);
        }
        printf("%d\n", memo[m]);
    }
}
```

Ex 24. *LISA - Pocket Money*

Resposta. Dada uma expressão aritmética com inteiros entre 0 e 9 e operadores $+$ e $*$, deve-se calcular um valor mínimo que pode ser obtido com uma parentização e o valor máximo que pode ser obtido por uma parentização. Parentizar uma expressão é, na verdade, escolher uma ordem para seus operadores. Assim, podemos definir a função f que recebe uma expressão e devolve uma parentização máxima, se soubermos calcular f sabemos calcular a expressão mínima analogamente.

Seja então a expressão s, como dito, $f(s)$ é o valor máximo de uma parentização. Podemos definir f da seguinte maneira, recursivamente.

- Se s é um inteiro (não tem nenhum operador), $f(s) = s$, pois esta é a única "parentização"possível.
- Caso contrário, devemos escolher um operador para ser calculado com prioridade sobre todos os outros nesta expressão. Todo operador divide a expressão em duas, para

calcular o valor obtido ao remover este operador, basta calcular o valor ótimo para as expressões obtidas. Conseguimos, assim, uma forma de calcular o valor de f a partir de partes de f.

Agora, devemos calcular a complexidade deste programa. A tabela de símbolos tem tamanho $O(n^2)$ e para calcular cada um dos estados realizamos trabalho $O(n)$. Assim a complexidade final fica $O(n^3)$.

Eu tive uma submissão aceita no SPOJ-BR, meu user lá é `victorsenam`. O link pro meu status neste problema é http://www.spoj.com/status/LISA,victorsenam/. Segue o código submetico no juíz, em C++.

```
#include <bits/stdc++.h>

using namespace std;
typedef unsigned long long int ull;
typedef long long int ll;

#ifndef ONLINE_JUDGE
#define DEBUG(...) {fprintf(stderr, __VA_ARGS__);}
#else
#define DEBUG(...) {}
#endif

const int N = 207;

ll memo[2][N][N];
int visi[2][N][N], turn;
int t, n;
char str[N];

ll pd (bool t, int i, int j) {
    if (i == j) {
        return str[i]-'0';
    }
    ll & me = memo[t][i][j];

    if (visi[t][i][j] == turn)
        return me;
    visi[t][i][j] = turn;

    if (t) me = LLONG_MAX;
    else me = 0;

    for (int k = i+1; k < j; k+=2) {
        ll loc = pd(t, i, k-1);
        if (str[k] == '+')
```

```
36              loc += pd(t, k+1, j);
37          else
38              loc *= pd(t, k+1, j);
39
40          if (t) me = min(me, loc);
41          else me = max(me, loc);
42      }
43      return me;
44  }
45
46  int main () {
47      scanf("%d", &t);
48      while (turn++ < t) {
49          scanf(" %s", str);
50          n = strlen(str);
51          printf("%lld %lld\n", pd(0, 0, n-1), pd(1, 0, n-1));
52      }
53  }
```

□