Algoritmos em Sequências

Notas de aula da disciplina TE: Técnicas de Construção de Algoritmos

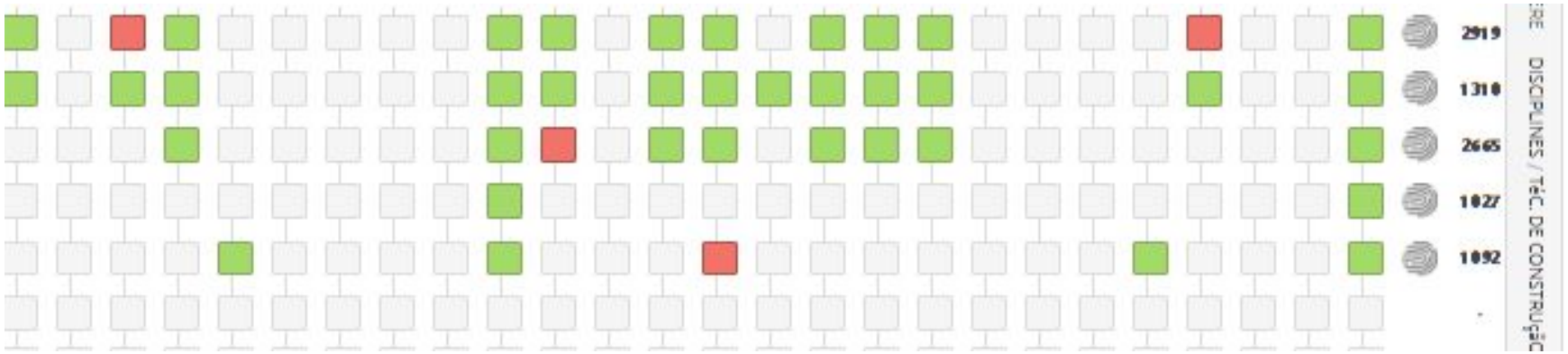
Fabiano de Souza Oliveira
(fabiano.oliveira@ime.uerj.br)

Paulo Eustáquio Duarte Pinto
(pauloedp arroba ime.uerj.br)

setembro/2020

TE: Técnicas de Construção de Algoritmos
Algoritmos em Sequências

Desempenho da semana passada



TE: Técnicas de Construção de Algoritmos
Algoritmos em Sequências

Problemas de 12/09/2020:

1341 - Crianças em uma Grade
2824 - Pudim
2842 - Dabriel e suas Strings
1833 - Decoração Natalina
1373 - Sequências DNA

1341 - Crianças em uma Grade

Contexto: É dada uma grade H x W (matriz) e o percurso de duas crianças, formando duas strings. Determinar o tamanho de cada string após a retirada da subsequência comum máxima.

Entrada: A primeira linha contém um inteiro t, o número de casos. Cada caso é descrito em várias linhas. A primeira contém H e W (1 ≤ H, W ≤ 20). Nas próximas H linhas vem a matriz de letras. Na linha seguinte 3 inteiros: N, x_0, y_0, o tamanho do percurso da primeira criança e sua posição inicial. Na próxima linha vem um string de tamanho N contendo o percurso da primeira criança. Depois informação semelhante para a segunda criança.

Saída: Para cada caso de teste, imprima o número do caso e o que sobra do percurso da primeira criança e depois o da segunda, quando se retira desses percursos a subsequência máxima comum.

Exemplo de entrada:

```
1
3  4
ABCD
DEFG
ABCD
4  1  1
EEES
3 3 1
NES
```

Exemplo de saída:

```
Case 1: 3 2
```

1341 - Crianças em uma Grade

Dicas:

1. A solução é construir os strings dos percursos e aplicar o algoritmo de Subsequência Máxima Comum.

2824 - Pudim

Contexto: Dados dois strings A e B, determinar o tamanho da subsequência crescente máxima.

Entrada: Um caso de teste, contendo dois strings A e B (1 ≤ |A|, |B| ≤ 5000), um em cada linha.

Saída: Imprimir o tamanho da subsequência consecutiva máxima.

Exemplo de entrada:

```
pudim
puhdim
```

Exemplo de saída:

```
5
```

2824 - Pudim

Dicas:

1. A solução é aplicar o algoritmo de Subsequência Máxima Comum.

2842 - Dabriel e suas Strings

Contexto: Dados dois strings A e B, determinar o tamanho da menor sequência que tem A e B como subsequências.

Entrada: Um caso de teste contendo as strings A e B ($1 \leq |A|, |B| \leq 1000$), uma em cada linha.

Saída: Para cada teste, imprimir T, o tamanho da menor sequência que tem A e B como subsequências.

Exemplo de entrada:

```
BOLA
BOTA
```

Exemplo de saída:

```
5
```

2824 - Pudim

Dicas:

1. A solução é aplicar o algoritmo de Subsequência Máxima Comum.

2. A sequência de tamanho mínimo tem tamanho $|A| + |B| - |SMC(A, B)|$.

1833 - Decoração Natalina

Contexto: São dadas duas sequências de lâmpadas coloridas e quer-se transformar a primeira na segunda, através de operações de Remoção (gasta 30 seg), Inserção(gasta 2,5 min) ou Troca(1 min), cada uma com ´tempo de execução diferente. É dado o custo por minuto para fazer o trabalho. Quer-se saber o custo mínimo para a operação desejada.

Entrada: A primeira linha contém o número de testes. Cada teste começa com um inteiro R ($1 \leq R \leq 10$), p custo do minuto, numa linha. Em seguida vem um inteiro M ($0 \leq M \leq 100$) com o tamanho da sequência errada; na próxima linha a sequência de cores; na linha seguinte, um inteiro N ($0 \leq N \leq 100$) com o tamanho da sequência desejada e, na última linha, as cores da sequência desejada. Há, no máximo, 20 cores e cada cor é um string de. no máximo, tamanho 50.

Saída: Para cada caso imprimir o número do caso e o custo mínimo.

Exemplo de entrada:

```
4
7
3
amarelo azul vermelho
4
ver,elho  verde  azul amarelo
```

Exemplo de saída:

```
Caso #1: R$ 31,50
```

1833 - Decoração Natalina

O problema é uma variante interessante da distância de edição, em que as operações de edição têm custos diferentes.

Pegadinha: N e M podem ser 0.

Trabalhar com custos inteiros (x 2) e só acertar na saída. A formatação da saída pode ser feita na "força bruta".

A leitura das strings é trabalhosa. Veja código no slide seguinte.

1833 - Decoração Natalina

```
void PreencheA(){
    string s, s2;    int i, j=0, k, l;
    cin.ignore();
    getline(cin, s);  i = 0;  l = s.length();    ncor=0;
    while(i< l){
        s2="";
        while(i < l && s[i] == ' ' ) i++;
         while(i < l && s[i] != ' '){
             s2 = s2+s[i]; i++;
         }
        for (k=1; k<=ncor; k++)
            if (s2 == cor[k]){
                A[++j] = char('A'+k-1);
                break;
            }
        if (k > ncor){
            cor[++ncor] = s2;
            A[++j] = char('A'+ncor-1);
        }
    }
}
```

1373 - Sequências DNA

Contexto: Trata-se de uma variante do problema clássico de subsequência máxima comum. Nesta variante é dado um valor k tal que as subsequências comuns têm que ser formadas por pedaços contíguos de tamanho no mínimo k.

Entrada: Vários casos de teste, iniciados por um inteiro K (1 ≤ K ≤ 100). Em seguida vêm duas linhas com um string cada de tamanho máximo = 1000. A linha com K=0 termina a entrada e não é processada.

Saída: O tamanho da maior subsequência comum entre os strings dados, levando em consideração a restrição estabelecida.

Exemplo de entrada:

```
3
lovxxelyxxxxx
xxxxxxxlovely
3
caaacac
caacac
```

Exemplo de saída:

```
6
6
```

1092 - Maior Subsequência Crescente

A questão é como preencher a posição [i,j] da matriz de distância de edição, quando A[i] = B[j]

	X	X	X	X	X	X	X	L	O	V	E	L	Y
L	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0
V	0	0	0	0	0	0	0	0	0	3	3	3	3
X	0	0	0	0	0	0	0	0	0	3	3	3	3
X	0	0	0	0	0	0	0	0	0	3	3	3	3
E	0	0	0	0	0	0	0	0	0	3	3	3	3
L	0	0	0	0	0	0	0	0	0	3	3	3	3
Y	0	0	0	0	0	0	0	0	0	3	3	3	6
X	0	0	0	0	0	0	0	0	0	3	3	3	6
X	0	0	0	0	0	0	0	0	0	3	3	3	6
X	0	0	3	3	3	3	3	3	3	3	3	3	6
X	0	0	3	4	4	4	4	4	4	4	4	4	6
X	0	0	3	4	5	5	5	5	5	5	5	5	6

1092 - Maior Subsequência Crescente

A questão é como preencher a posição [i,j] da matriz de distância de edição, quando A[i] = B[j]

	C	A	A	C	A	C
C	0	0	0	0	0	0
A	0	0	0	0	0	0
A	0	0	3	3	3	3
A	0	0	3	3	3	3
C	0	0	3	3	3	3
A	0	0	3	3	4	4
C	0	0	3	3	4	?

1092 - Maior Subsequência Crescente

A questão é como preencher a posição [i,j] da matriz de distância de edição, quando A[i] = B[j]

	C	A	A	C	A	C
C	0	0	0	0	0	0
A	0	0	0	0	0	0
A	0	0	3	3	3	3
A	0	0	3	3	3	3
C	0	0	3	3	3	3
A	0	0	3	3	4	4
C	0	0	3	3	4	6

Quando A[i]=B[j], deve-se procurar qual o tamanho R da maior sequência consecutiva comum. Então se R > k, tem que se examinar todas as posições T[i-l][j-l] e adotar o máximo T[i-l][j-l]+l para $k \leq l \leq R$.

FIM