Algoritmos em Sequências

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

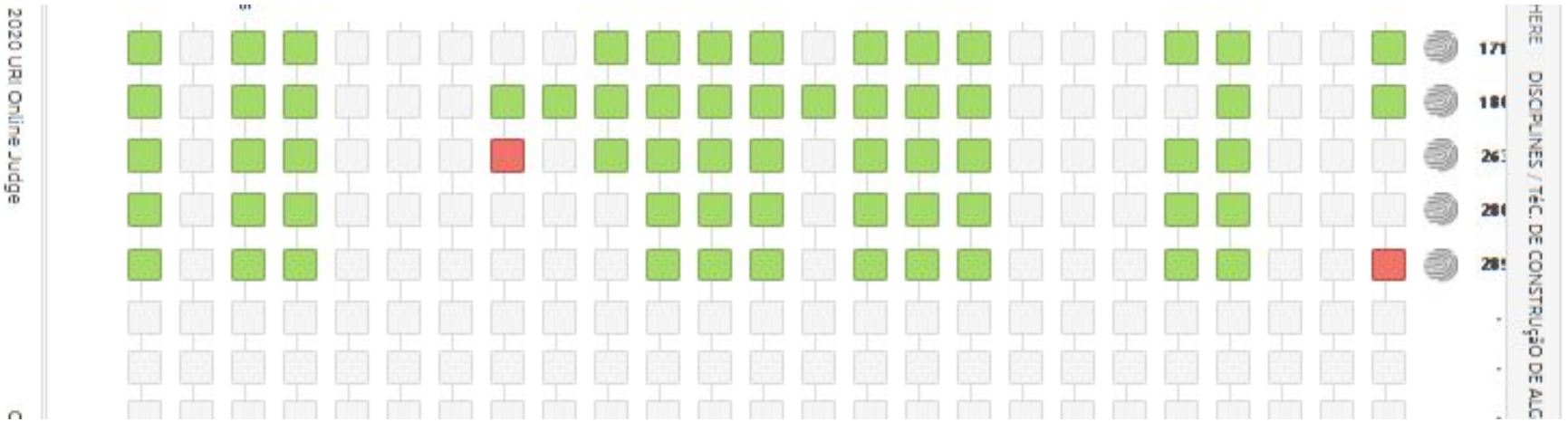
Fabiano de Souza Oliveira
(fabiano.oliveira@ime.uerj.br)

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 05/09/2020:

1027 - Onda Crítica
1310 - Lucro
2919 - Melhor Ordem
2665 - Hipercampo
1092 - Maior Subsequência Crescente

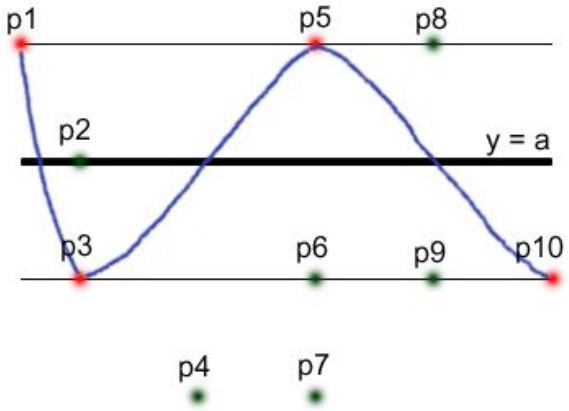
1027 - Onda Crítica

Contexto: Determinar o maior número de pontos que forma uma "onda crítica": supondo uma linha imaginária y = a, para alguma constante a, todos os pontos na onda crítica deverão ter diferentes coordenadas x, e suas coordenadas y devem, alternadamente, ser na forma a-1 ou a+1.

Entrada: Haverá vários casos de testes. Cada caso inicia com o número de pontos N ≤ 1000, seguido de N linhas com as coordenadas inteiras x e y de cada ponto.

Saída: Para cada caso de teste, imprima o número máximo de pontos que podem ser incluídos em uma única "onda crítica" qualquer.

Exemplo de entrada:



Exemplo de saída:

4

1027 - Onda Crítica

Dicas:

1. **A solução é aplicar o algoritmo de Sequência Crescente Máxima, adaptada para o problema:**

a. Ordenar os pontos por suas coordenadas (x,y), lexicograficamente

b. Guardar, para cada ponto p_i, o tamanho das maiores subsequências C(i) e B(i) de pontos em uma "onda crítica" que terminam em p_i, nos seguintes casos:
   i. p_i está no eixo de cima da curva (C(i))
   ii. p_i está no eixo de baixo da curva (B(i))

1027 - Onda Crítica

Dicas:

2. **Empregar a versão $O(n^2)$ do algoritmo:**

a. $C(i) = \text{máx} \{ B(j) + 1 \mid j < i \text{ , } p_j.x < p_i.x \text{ , } p_j.y = p_i.y-2 \}$

$B(i) = \text{máx} \{ C(j) + 1 \mid j < i \text{ , } p_j.x < p_i.x \text{ , } p_j.y = p_i.y+2 \}$

b. Note que pelo fato de que nem todo ponto $p_j < p_i$ deve ser considerado para determinar C(i), B(i), a versão mais eficiente que emprega busca binária é inadequada

3. **Solução:** $\text{máx} \{ B(i), C(i) \mid 1 \leq i \leq N \}$

1310 - Lucro

Contexto: George tem um circo e quer avaliar se leva o circo para certa cidade para um período de N dias. Ele tem um custo fixo diário D e tem as previsões de receitas para o período $R_1 \ldots R_N$. Ele quer saber qual o lucro máximo que ele pode ter nesse período.

Entrada: Vários casos de teste, terminados por fim de arquivo. A primeira linha contém um inteiro N ($1 \leq N \leq 50$), o número de dias que George pode ficar na cidade. A segunda linha contém um inteiro D ($0 \leq D < 1000$), o custo diário. A seguir vêm N linhas contendo um inteiro cada, $R_1 \ldots R_N$ ($0 \leq R_i < 1000$), as receitas diárias previstas.

Saída: Para cada teste imprimir o lucro máximo que George pode ter.

Exemplo de entrada:

```
6
20
18
35
6
80
15
21
```

Exemplo de saída:

```
61
```

1310 - Lucro

Dicas:

1. **O lucro diário é expresso pela sequência:**

   $(R_1-D)\ (R_2-D)\ldots(R_N-D)$.

2. **A solução é aplicar o algoritmo de Sequência Consecutiva de Soma Máxima.**

2919 - Melhor Ordem

Contexto: É descrita uma brincadeira que consiste em determinar o tamanho da Susequência Crescente Máxima a partir de uma sequência de inteiros com N números.

Entrada: Vários casos de teste terminados por fim de arquivo. Cada caso é descrito em 3 linhas. A primeira contém um inteiro N ($1 \leq N \leq 5.10^5$), o tamanho da sequência dada. Na segunda linha vem N inteiros com valores limitados a 10^9.

Saída: Para cada teste, imprimir T, o tamanho da subsequência crescente máxima.

Exemplo de entrada:

```
5
2 5 3 45 67
8
5 7 6 4 12 8 9 10
```

Exemplo de saída:

```
4
5
```

2919 - Melhor Ordem

Dicas:

1. Problema clássico de SCM (subsequência crescente máxima)

2. Observação sobre a implementação do algoritmo dado:

   j ← BuscaBinária (T, 1, k, S[i])

   A busca binária deve retornar o índice j do menor elemento de T, tal que T[j] ≥ S[i]

3. **Em C++ pode-se usar a função lower_bound**

   ```
   j = lower_bound( T+1, T + k, S[i] ) - T;
   ```

   Observar que se (S[i] > T[k]) a função retorna k.
   Isso não é problema para nosso algoritmo.

2919 - Melhor Ordem

Algoritmo em C para BuscaBinária:

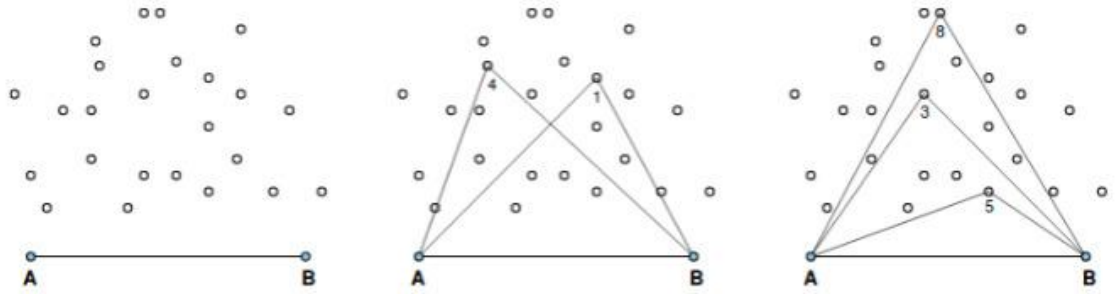
j ← BuscaBinária (T, 1, k, S[i])

```
int low_bound(int low, int high, int e){
      if ( low < 0) return 0;
      if (low >= high ){
            if ( e <= T[low] ) return low;
            return low+1;
      }
      int mid=(low+high)/2;
      if ( e > T[mid])
            return low_bound(mid+1,high,e);
      return low_bound(low,mid,e);
}
```

Tal como no caso anterior, a função retorna o menor índice j de T tal que T[j] ≥ S[i]. entretanto, se (S[i] > T[k]) a função retorna k+1. Isso não é problema no atual algoritmo. T é global.

2665 - Hipercampo

Contexto: Dadas duas âncoras A e B e N pontos, determinar o número máximo de triângulos que podem estar encaixados em sequência, cada um só interceptando com AB.



Entrada: Um caso de teste descrito em várias linhas. A primeira linha contém 3 inteiros N ($1 \leq N \leq 100$), A e B ($0 < A < B \leq 10^4$), os pontos âncora. A seguir vêm N linhas com 2 inteiros X_i Y_i, ($0 < X_i, Y_i \leq 10^4$), as coordenadas dos pontos.

Saída: O número máximo de triângulos que podem ser formados.

Exemplo de entrada:

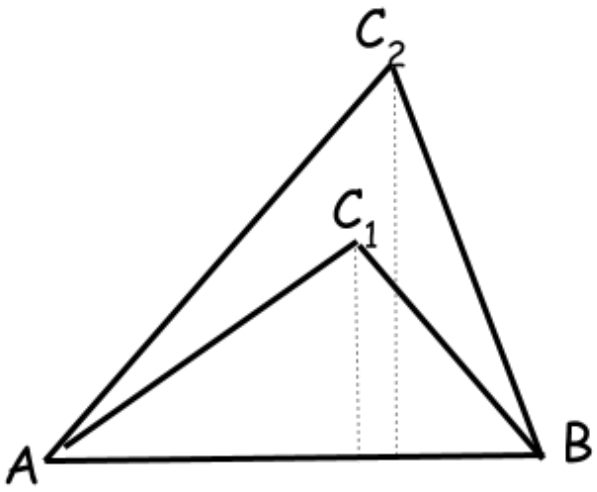
```
4  1  10
2  4
5  1
6  5
7  8
```

Exemplo de saída:

```
3
```

2665 - Hipercampo

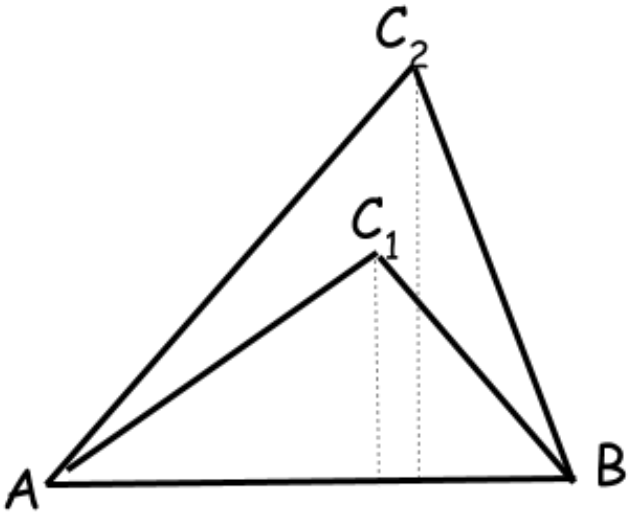
Condição para um triângulo estar interior ao outro



Para que o triângulo AC_1B seja interior ao triângulo AC_2B, é necessário que tenhamos: ângulo ABC_1 < ângulo ABC_2 e ângulo BAC_1 < ângulo BAC_2. O ângulo pode ser representado por seu arcocosseno

2665 - Hipercampo

Modelagem como Sequência Crescente Máxima:



Se ordenarmos os pontos pelo arcocosseno do ângulo no ponto A do triângulo BAC_i de cada ponto p_i, basta agora aplicar o algoritmo de SCM usando como critério de comparação o arcocosseno do ângulo ABC_i.

1092 - Maior Subsequência Crescente

Contexto: É dada uma matriz de inteiros. Quer-se saber a submatriz com mais elementos tal que, se linearizada, é uma sequência crescente.

1	**2**	**5**
4	**6**	**7**
10	8	3

1	**2**	1	2
9	**6**	7	3
8	**7**	2	8

22	2	14	22	23
16	**21**	**22**	**31**	31
57	**33**	**43**	**45**	50
46	**51**	**66**	**83**	93

Entrada: Vários casos de teste, terminados por 0 0, que não deve ser processado. Para cada teste, a primeira linha contém N e M, ($1 \leq M,N \leq 600$), as dimensões da matriz. A seguir vêm N linhas com M inteiros em cada uma, os elementos da matriz.

Saída: Um inteiro o número da matriz máxima encontrada.

Exemplo de entrada:

```
3 5
1 4  2 15 1
8 8  7 10 4
20 12 11 15 6
```

Exemplo de saída:

```
4
```

1092 - Maior Subsequência Crescente

Dica: Problema com solução ad-hoc (específica).

1	4	2	15	1
8	8	7	10	4
20	12	11	15	6

1. Inicialmente, obter as soluções em uma linha ou coluna

2	1	2	1	1
1	1	2	1	1
1	1	2	1	1

3	3	3	1	3
2	2	2	2	2
1	1	1	1	1

2. Processar as matrizes H horizontal e V vertical auxiliares

1092 - Maior Subsequência Crescente

1	4	2	15	1
8	8	7	10	4
20	12	11	15	6

2	1	2	1	1
1	1	2	1	1
1	1	2	1	1

3	3	3	1	3
2	2	2	2	2
1	1	1	1	1

2. Processar as matrizes H horizontal e V vertical auxiliares

O processamento consiste em varrer a matriz horizontal H. Para cada (i,j), desce-se na coluna j até i+V[i,j]-1, checando o máximo retângulo possível.

2	1	2	1	1
1	1	2	1	1
1	1	2	1	1

Max. retângulo = 2 x 1

Max. retângulo = 1 x 2

Max. retângulo = 1 x 3

1092 - Maior Subsequência Crescente

1	4	2	15	1
8	8	7	10	4
20	12	11	15	6

2	1	2	1	1
1	1	2	1	1
1	1	2	1	1

3	3	3	1	3
2	2	2	2	2
1	1	1	1	1

2. Processar as matrizes H horizontal e V vertical auxiliares

O processamento consiste em varrer a matriz horizontal H. Para cada (i,j), desce-se na coluna j até i+V[i,j]-1, checando o máximo retângulo possível.

2	1	2	1	1
1	1	2	1	1
1	1	2	1	1

Max. retângulo = 2 x 1

Max. retângulo = 2 x 2

FIM