Tópicos Avançados em Algoritmos

Hamilton José Brumatto

Bacharelado em Ciências da Computação - UESC

25 de abril de 2019

Árvores de Segmentos

O problema

- Vamos considerar um vetor de valores, com índices associados aos valores:

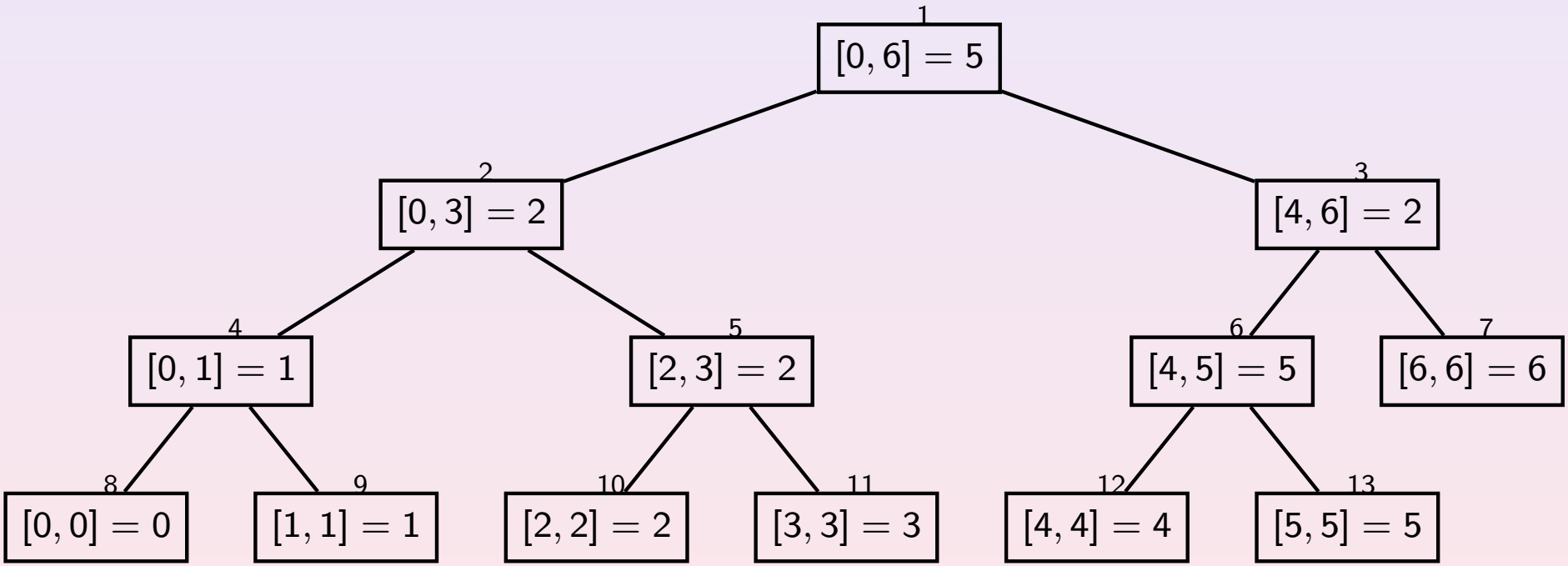
Vetor	Valores	18	17	13	19	15	11	20
A	Índices	0	1	2	3	4	5	6

- Queremos saber qual o menor valor entre 2 determinados índices (RMQ - *Range Minimum Query*):
  - $RMQ(1, 3) = 2$
  - $RMQ(0, 1) = 1$
  - $RMQ(0, 6) = 5$
- Um algoritmo trivial é uma busca no vetor, do índice i a j, teremos um custo $O(n)$ por consulta.

A Árvore de Segmentos

- Vamos organizar a árvore de segmentos, como uma árvore binária na forma de um Heap.
- O Heap é uma árvore binária organizada na forma de um vetor. Para o nó de índice i, os filhos estão no índice $2 * i$ e $2 * i + 1$. Considerando o primeiro índice 1.
- O raiz da árvore, para o exemplo acima, seria o índice do $RMQ(0, 6)$, as folhas seriam os valores individuais com o índice do $RMQ(i, i)$ que é i
- O tamanho da árvore seria $2 \times 2^{\lfloor(\log_2(n))\rfloor+1}$. Desta forma podemos considerar a complexidade no espaço $O(4n) = O(n)$

A Árvore de Segmentos



Busca do *RMQ* na árvore

- Vamos buscar $RMQ(1,3)$:
  - O raiz tem um intervalo $(0,6)$ maior então precisamos ver os filhos:
  - $(0,6) \rightarrow$ Filho esquerdo $(0,3)$ também com intervalo maior, precisamos ver seus filhos. O filho direito $(4,6)$ não precisa ser consultado.
  - $(0,3) \rightarrow$ Filho esquerdo $(0,1)$ também tem valores fora, filho direito $(2,3)$ está dentro e podemos pegar seu resultado: 2.
  - $(0,1) \rightarrow$ Filho esquerdo $(0,0)$ está todo fora, filho direito $(1,1)$ dentro, pegamos: 1, que retornamos para o nó $(0,1)$.
  - Para o nó $(0,3)$ ficamos com $min\{(0,1) : 1(17), (2,3) : 2(13)\} = 2$
  - Voltamos para o raiz com o resultado $2 \rightarrow A[2] = 13$.
- Assim uma pesquisa *RMQ* pode ser realizada em $O(\log n)$

Atualizando a árvore

- Vamos supor que há uma mudança em algum valor do vetor, por exemplo: $A[5] = 99$. A atualização é feita da folha para o raiz:
  - A folha $[5, 5]$, índice 13 não se altera.
  - No ramo $[4, 5]$, índice 6, comparamos ambos filhos, 5 e 4, $A[4] < A[5]$ logo atualizamos o ramo para 4.
  - No ramo $[4, 6]$, idem: 4 e 6: $A[4] < A[6]$, e atualizamos para 4.
  - No raiz $[0, 6]$, idem: 2 e 4: $A[2] < A[4]$, atualizamos para 2.

Classe SegmentTree - privates

```
class SegmentTree {
private:  vector<int> st, A;
 int n;
 int left (int p) { return p << 1; }
 int right(int p) { return (p << 1) + 1; }
 void build(int p, int L, int R) {   if (L == R) st[p] = L;
  else {
   build(left(p) , L , (L + R) / 2);
   build(right(p), (L + R) / 2 + 1, R);
   int p1 = st[left(p)], p2 = st[right(p)];
   st[p] = (A[p1] <= A[p2]) ?  p1 :  p2;
  }
 }
 int rmq(int p, int L, int R, int i, int j) {
  if (i > R || j < L) return -1; // fora
  if (L >= i && R <= j) return st[p]; // dentro
  int p1 = rmq(left(p), L, (L+R)/2, i, j);
  int p2 = rmq(right(p), (L+R)/2 + 1, R, i, j);
  if (p1 == -1) return p2;
  if (p2 == -1) return p1;
  return (A[p1] <= A[p2]) ?  p1 :  p2;
 }
```

Classe SegmentTree - publics

```
public:
  SegmentTree(const vi &_A) {
    A = _A;
    n = (int)A.size();
    st.assign(4 * n, 0);
    build(1, 0, n - 1);
  }
  int rmq(int i, int j) {
    return rmq(1, 0, n - 1, i, j);
    }
};
```

URI - 2531 - Compras em FdI

Está chegando a grande final do Campeonato Nlogonense de Surf Aquático. Ano que vem, a final ocorrerá na cidade de Foça do Iguachim (FdI)! A região de FdI e das cidades próximas é famosa por seu comércio, composto por diversas lojas que costumam vender diversos produtos a preços mais atraentes que no restante do país. Você quer aproveitar a viagem para FdI para comprar o novo celular Aifôni (R) (Na verdade, você queria um Sãosunga (R), mas este celular é um verdadeiro estouro!)!

Existem **N** lojas na região, numeradas de 1 a **N**. Todas as lojas vendem o celular, embora o preço do aparelho pode ser diferente em cada loja. Para não tornar sua viagem cansativa, você pode considerar não visitar todas as **N** lojas, mas sim visitar apenas as lojas entre duas dadas lojas **i** e **j**, inclusive. Você está interessado na maior diferença de preços do aparelho entre as lojas visitadas. A diferença é dada por $|\mathbf{M}-\mathbf{m}|$, onde **M** é o maior preço dentre as lojas visitadas, e **m** é o menor.

Além disso, as lojas podem alterar o preço do celular como desejarem! Sua tarefa é determinar, para várias consultas, a maior diferença de preços nas lojas entre duas dadas lojas, considerando também eventuais alterações de preços nas lojas.