Busca Binária

Wladimir Araújo Tavares [1]

[1]Universidade Federal do Ceará - Campus de Quixadá

24 de abril de 2018

1 Pão a metro

2 FACVSPOW - Factorial vs Power

3 Maior subsequencia crescente

4 SUBSUMS

5 2SUM

6 3SUM

Pão a metro

Dado dois inteiro n e k e um vetor de inteiros de tamanho k $(s_1, \ldots, s_k)$.
Encontre o maior inteiro L tal que $\sum_{i=1}^{K} \lfloor \frac{s_1}{L} \rfloor \geq n$.
Entrada:

```
10
4
120  89  230  177
```

Por exemplo,
para L = 50 temos que

$$\lfloor \frac{120}{50} \rfloor + \lfloor \frac{89}{50} \rfloor + \lfloor \frac{230}{50} \rfloor + \lfloor \frac{177}{50} \rfloor = 2 + 1 + 4 + 3 = 10 \quad (1)$$

para L = 60 temos que

$$\lfloor \frac{120}{60} \rfloor + \lfloor \frac{89}{60} \rfloor + \lfloor \frac{230}{60} \rfloor + \lfloor \frac{177}{60} \rfloor = 2 + 1 + 3 + 2 = 8 \quad (2)$$

Pão a metro

Observações:

1. Se $\sum_{i=1}^{K} \lfloor \frac{s_1}{L} \rfloor \geq n$ e $L' \leq L'$ então $\sum_{i=1}^{K} \lfloor \frac{s_1}{L'} \rfloor \geq n$
2. Se $\sum_{i=1}^{K} \lfloor \frac{s_1}{L} \rfloor < n$ e $L' \geq L$ então $\sum_{i=1}^{K} \lfloor \frac{s_1}{L'} \rfloor < n$

Essas observações tornam possíveis a utilização da busca binária.

Pão a metro

Algorithm 1 Pão a metro

function PaoMetro$(n, (s_1, \ldots, s_k))$
    $inicio \leftarrow 1$
    $fim \leftarrow \max\{s_1, \ldots, s_k\}$
    $sol \leftarrow -1$
    while $inicio < fim$ **do**
        $meio \leftarrow inicio + \lfloor (fim - inicio)/2 \rfloor$ ▷ Evita overflow
        if $\sum_{i=1}^{K} \lfloor \frac{s_1}{meio} \rfloor >= n$ **then**
            $sol \leftarrow meio$
            $inicio \leftarrow meio + 1$
        else
            $fim \leftarrow meio - 1$

1 Pão a metro

2 FACVSPOW - Factorial vs Power

3 Maior subsequencia crescente

4 SUBSUMS

5 2SUM

6 3SUM

FACVSPOW - Factorial vs Power

Considere a função $f(n) = n!$ e $g(n) = a^n$, onde n é um inteiro positivo.
Para qualquer inteiro positivo $a > 1$, $f(n) > g(n)$ para todo $n \geq k$.
Encontre o menor inteiro positivo n tal que $f(n) > g(n)$, para um dado inteiro positivo.
Por exemplo, para a $=$ 2, n $=$ 4. $24 = 4! > 2^4 = 16$
Por exemplo, para a $=$ 3, n $=$ 7. $5040 = 7! > 3^7 = 2187$
Por exemplo, para a $=$ 4, n $=$ 9. $362880 = 9! > 4^9 = 262144$

FACVSPOW - Factorial vs Power

Primeiramente temos que utilizar um método para resolver o problema de calcular valores muito grandes de $f(n) = n!$ e $g(n) = a^n$ sem causar overflow. Vamos utilizar a função logarítmica que consegue transformar multiplicações em adições e divisões em subtrações. A seguir, vamos aplicar essa propriedade matemática para $f(n)$ e $g(n)$:

$$f(n) = n! = log(n!) = \sum_{i=1}^{n} log(i) \tag{3}$$

$$g(n) = a^n = log(a^n) = n * log(a) \tag{4}$$

1 Pão a metro

2 FACVSPOW - Factorial vs Power

3 Maior subsequencia crescente

4 SUBSUMS

5 2SUM

6 3SUM

Maior subsequencia crescente

Dada um sequencia s qualquer, encontre o tamanho da maior subsequência crescente de s. Lembrando que s é uma subsequência de v se s é obtida pela remoção de alguns elementos de v.
Por exemplo, $s' = (3, 4, 5, 7)$ é uma subsequencia de $s = (3, 4, 3, 5, 2, 7)$

Maior subsequencia crescente

Idéia do algoritmo: Dado um vetor A, para cada $A[1 \ldots i]$ guarde uma sequência crescente com o mesmo tamanho da maior subsequência de $A[1 \ldots i]$

Maior subsequencia crescente

```
//lower_bound
int busca_binaria(vector<int> &seq, int x){
  int lo = 0;int hi = seq.size() - 1;
  int ret = -1;
  while( lo <= hi ){
      int mid = lo + (hi-lo)/2;
      if( seq[mid] >= x ){
        ret = mid; hi  = mid-1;
      }else{
        lo = mid+1;
      }
  }
  return ret;
}
int lis(vector<int> &v){
  vector<int> seq;
  for(int i=0; i<v.size(); i++){
    int ret = busca_binaria(seq, v[i]);
    if(ret == -1) seq.push_back(v[i]);
    else seq[ret] = v[i];
  }
  return seq.size();
}
```

lower_bound

Returns an iterator pointing to the first element in the range [first,last) which does not compare less than val.

```
template <class ForwardIterator, class T>
ForwardIterator lower_bound (ForwardIterator first, ForwardItera
{
  ForwardIterator it;
  iterator_traits<ForwardIterator>::difference_type count, step;
  count = distance(first,last);
  while (count>0)
  {
    it = first; step=count/2; advance (it,step);
    if (*it<val) {// or: if (comp(*it,val)), for version (2)
      first=++it;
      count-=step+1;
    }
    else count=step;
  }
  return first;
}
```

Maior subsequencia crescente

```
int lis(vector<int> &v){
  vector<int> seq;
  for(int i=0; i<v.size(); i++){
    vector<int>::iterator it = lower_bound(seq.begin(), seq.end(
    if(it == seq.end()) seq.push_back(v[i]);
    else *it = v[i];
  }
  return seq.size();
}
```

Maior subsequencia crescente

Uma cadeia de caracteres é uma sequência de letras do alfabeto. Uma cadeia de caracteres crescente é uma sequência de letras onde a próxima letra (da esquerda para a direita) nunca ocorre antes no alfabeto do que a letra anterior. Por exemplo ABBD é crescente, enquanto ABBAD não é crescente.

Uma subsequência de uma cadeia de caracteres é uma cadeia de caracteres que pode ser obtida a partir da remoção de zero ou mais caracteres da cadeia de caracteres original. Por exemplo ANNA é uma subsequência de BANANAS. Outro exemplo seria ANNS, que é uma subsequência crescente de BANANAS.

Dada uma cadeia de caracteres S , escreva um programa para determinar o tamanho da maior subsequência de S que é uma cadeia de caracteres crescente.

Maior subsequencia crescente

```
//upper_bound
int busca_binaria(vector<int> &seq, int x){
  int lo = 0; int hi = seq.size() - 1;
  int ret = -1;
  while( lo <= hi ){
    int mid = lo + (hi-lo)/2;
    if( seq[mid] > x ){
        ret = mid; hi  = mid-1;
    }else{
        lo = mid+1;
    }
  }
  return ret;
}
int lis(string s){
  vector<int> seq;
  for(int i=0; i<s.size(); i++){
    int ret = busca_binaria(seq, s[i]);
    if(ret == -1) seq.push_back(s[i]);
    else seq[ret] = s[i];
  }
  return seq.size();
}
```

Maior subsequencia crescente

AAXBBXZZX

```
char  A ret -1
char  A ret -1
char  X ret -1
char  B ret 2
char  B ret -1
char  X ret -1
char  Z ret -1
char  Z ret -1
char  X ret 5
```

7

1 Pão a metro

2 FACVSPOW - Factorial vs Power

3 Maior subsequencia crescente

4 SUBSUMS

5 2SUM

6 3SUM

SUBSUMS

Dada uma sequencia de N ($1 \leq N \leq 34$) números $S_1, \ldots, S_N$ ($-20.000.000 \leq S_i \leq 20.000.000$), determine quantos subconjuntos de S (incluindo o vazio) têm uma soma entre A e B ($-500.000.000 \leq A \leq B \leq 500.000.000$).

Entrada:

$3 \; -1 \; 2$
1
-2
3

Saída: 5
Os 5 subconjuntos com soma entre -1 e 2:

1. $0 = 0$ (o subconjunto vazio)
2. $1 = 1$
3. $1 + (-2) = -1$
4. $-2 + 3 = 1$
5. $1 + (-2) + 3 = 2$

SUBSUMS

Desenvolva um algoritmo $O(n * 2^{n/2})$.

1. Divida o conjunto em duas metades
2. Para cada metade, construa todos os subconjuntos possiveis e guarde a soma de cada subconjunto.
3. Para cada subconjunto de uma metade, encontre quantos subconjuntos da outra metade que a soma dos dois subconjuntos está entre a e b. (Use a busca binária)

1 Pão a metro

2 FACVSPOW - Factorial vs Power

3 Maior subsequencia crescente

4 SUBSUMS

5 2SUM

6 3SUM

2SUM

Seja L um vetor de inteiros. Encontre dois elementos distintos x e y em L, tal que $x + y = K$.
Complexidade: $O(nlogn)$
Problema: `http://www.codcad.com/problem/53`

1 Pão a metro

2 FACVSPOW - Factorial vs Power

3 Maior subsequencia crescente

4 SUBSUMS

5 2SUM

6 3SUM

3SUM

Seja L um vetor de inteiros. Encontre dois elementos distintos x, y e z em L, tal que $x + y + z = K$.
Complexidade: $O(n^2 logn)$
Problema: `http://www.spoj.com/problems/SUMFOUR/`