Universidade Federal do Ceará

Campus de Quixadá

QXD0153 - Desafios de Programação

Lista 3 - Estruturas de Dados

1. Implemente uma pilha de inteiros que tem a operação minimo() que devolve o menor inteiro da pilha em tempo constante O(1) mantendo a complexidade de tempo das outras operações.

   Solução Raul:

```
struct MinStack{
    stack<pair<int, int>> S;
    void push(int elem){
        if(S.size() == 0){
            S.push(make_pair(elem, elem));
        }else{
            int menor_atual = S.top().second;
            if(elem < menor_atual){
                S.push(make_pair(elem, elem));
            }else{
                S.push(make_pair(elem, menor_atual));
            }
        }
    }
    int top(){
        return S.top().first;
    }
    void pop(){
        S.pop();
    }
    int minimo(){
        return S.top().second;
    }
    bool empty(){
```

```
        return S.empty();
    }
};
```

2. Desenvolva um algoritmo de complexidade $O(n)$ que dado um vetor de inteiros com elementos repetidos, encontre a soma de todos os elementos distintos no vetor.

   Solução Wallison:

```
int somar(vector<int> v) {
    unordered_map<int, int> mapa;
    int soma = 0;
    for(int x : v) {
        if(mapa.find(x) == mapa.end()) {
            mapa[x] = x;
            soma += x;
        }
    }
    return soma;
}
```

3. Desenvolva um algoritmo de complexidade $O(n^2)$ que dado um vetor de inteiros distintos, encontre dois pares (x, y) e (z, w) tal que $xy = zw$, onde x, y, z e w são elementos distintos.

   Solução Daiane:

```
void findElems(vector<int> V, int& x, int& y, int& z, int& w)
{

    unordered_map<int, pair<int, int>> map;

    for(auto v : V){
        for(auto u : V){
            if(u != v){
                if(map.count(u*v) == 0){
```

```
                map[u*v] = make_pair(u,v);
            }else{
                if(u != map[u*v].second && v != map[u*v].first){
                    x = u;
                    y = v;
                    z = map[u*v].first;
                    w = map[u*v].second;
                }
            }
        }
    }
}

}
```

4. Suponha que você tem dois animais de estimação e você ama ambos muito. Você vai a uma loja de animal de estimação para comprar artigos diferentes para seus animais de estimação. Mas você pergunta ao vendedor apenas os artigos que estão realmente em pares. Nesta loja, os artigos são identificados por números inteiros. Então você que contar o número de artigos que você pode comprar para seus animais de estimação.

   A primeira linha da entrada contém um inteiro N representando o número de artigos da loja. A segunda linha contém N inteiros separados que descrevem os artigos da loja.

Entrada	Saída
7	6
10 10 10 20 20 10 20	

   Posso comprar 6 pares de produtos (10,10),(10,10),(20,20).

   Solução João Vitor:

```
map<int, int> m;

for(unsigned int i=0; i<v.size(); i++ ){
    m[v[i]]++;
```

```
}

int count = 0;
for(map<int,int>::iterator i = m.begin(); i!=m.end(); i++ ){
    count+=((i->second)/2);
}

count*= 2;

cout<<count<<endl;
```

5. Desenvolva um algoritmo que dado um vetor de dígitos (valores de 0 até 9), encontre a menor soma possível de dois números formados a partir dos dígitos do vetor. Todos os dígitos do vetor devem ser usados para forma os dois números.

   Entrada: [7,8,4,5,2,3]

   Saída: 605

   A soma mínima é formada pela soma dos números 358 e 247.

   Solução Wallison:

```
priority_queue<int, vector<int>, greater<int> > min_heap;
int n;
cin >> n;
for (int i = 0; i < n; ++i) {
        int j;
        cin >> j;
        min_heap.push(j);
}
int x = 0, y = 0;
while(!min_heap.empty()) {
        if (!min_heap.empty()) {
                int x_ = min_heap.top();
                min_heap.pop();
                x = 10*x + x_;
        }
```

```
        if (!min_heap.empty()) {
                int y_ = min_heap.top();
                min_heap.pop();
                y = 10*y + y_;
        }



}
cout << x+y << endl;
```

Fonte: `http://practice.geeksforgeeks.org/problems/min-sum-formed-by-digits/0`

6. Dado n cordas de diferentes tamanhos, nós precisamos conectá-las em uma única corda. O custo para conectar duas cordas é igual a soma dos seus comprimentos. Desenvolva um algoritmo para conectar n cordas com o custo mínimo.

   Entrada: [4,3,2,6]

   Saída: 29

   Solução Pedro Olimpio:

```
int vet[100000];
scanf("%d", &n);
for (int i = 0; i < n; i++){
        scanf("%d", vet + i);
}
priority_queue<int, vector <int>, greater<int> > pq(vet, vet + n);
int cont = 0;
while (pq.size() > 1){
        x = pq.top();
        pq.pop();
        y = pq.top();
        pq.pop();
        cont += x + y;
        pq.push(x + y);
}
```

```
printf("%d\n", cont);
```

7. Dado um fluxo (=stream) de n números inteiros. Considere o problema de inserir um inteiro no fluxo e imprimir a mediana do fluxo formado pela inserção desse inteiro. A mediana deve ser encontrada de maneira incremental, ou seja, online. Se o tamanho do vetor é ímpar, então a mediana é elemento do meio do vetor depois de ordenado. Se o tamanho do vetor é par, então a mediana é a média dos dois valores do meio do vetor depois de ordenado. Desenvolva um algoritmo de complexidade $O(nlogn)$ para encontrar a mediana de um fluxo de n inteiros de maneira incremental.

   Entrada: n=4 v = [5,15,1,3]

   Saída [5,10,5,4]

   Solução Ana Paula:

```
priority_queue<int, vector<int>, greater<int> > min;
priority_queue<int, vector<int>, less<int> > max;

int n, aux, m = 0;

cin >> n;


for(int i = 0; i < n; i++){
        cin >> aux ;
        if (aux > m){
                if (min.size() > max.size()){
                        max.push(min.top());
                        min.pop();
                }
                min.push(aux);
        } else {
                if (max.size() > min.size()){
                        min.push(max.top());
                        max.pop();
```

```
            }
            max.push(aux);
        }



        if (min.size() == max.size())
            m = (min.top() + max.top()) / 2;
        else {
            m = min.size() > max.size() ? min.top() : max.top() ;
        }
        //cout << "m:" << m << endl;
        medianas.push_back(m);
    }
    for (auto& elem : medianas){
        cout << elem << endl ;
    }
    cout << endl;
```

Fonte: `http://practice.geeksforgeeks.org/problems/find-median-in-a-stream/0`

8. Dado k vetores ordenados, sua tarefa é realizar a impressão do entrelaçamento("merge") dos vetores resultando em um vetor ordenado.

   (a) Desenvolva um algoritmo $O(n \lg n)$, onde n é o número total de elementos.

```
vector<int> merge_nlgn(vector< vector < int > > s, int k){
    vector<int> c;
    for (auto& a : s){
        c.insert(c.end(), a.begin(), a.end());
    }
    sort(c.begin(), c.end());
    return c;
}
```

   (b) Desenvolva um algoritmo $O(n \lg k)$, onde n é o número total de elementos.

```
vector<int> merge_nlgk(vector< vector< int > > v, int k){
  vector<int> c;
  priority_queue< pair<int,pair<int, int> >,
                  vector<pair<int,pair<int,int> > >,
                  ComparaPar > H;
  for (int i = 0; i < k; i++){
    H.push(make_pair(v[i][0], make_pair(i,0)));
  }
  pair<int, pair<int, int>> aux;
  int i, j;
  while (!H.empty()){
    aux = H.top(); H.pop();
        c.push_back(aux.first);
        i = aux.second.first;
        j = aux.second.second;
        if (j + 1 < v[i].size()){
          H.push(make_pair(v[i][j+1], make_pair(i, j+1)));
        }
  }
  return c;
}
```

9. Dado uma sequência $A[1], \ldots, A[n]$. Uma pergunta pode ser definida da seguinte maneira:

$$pergunta(i,j) = max\{\sum_{k=x}^{y} A[k] | i \leq x \leq y \leq j\}$$

Dado M perguntas, desenvolva um algoritmo que responde estas perguntas.

```
class SegmentTree {
private:
        vector <int> A;
        vector <int> sum;
```

```
vector <int> prefixSum;
vector <int> sufixSum;
vector <int> maxSum;

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){
    sum[p] = A[L];
        prefixSum[p] = A[L];
        sufixSum[p] = A[L];
        maxSum[p] = A[L];
  }else{
    build(left(p), L, (L+R)/2);
        build(right(p), (L+R)/2+1, R);
int p1 = left(p);
        int p2 = right(p);
        sum[p] = sum[p1] + sum[p2];
        prefixSum[p] = max(prefixSum[p1], sum[p1]+prefixSum[p2]);
        sufixSum[p] = max(sufixSum[p2], sum[p2]+sufixSum[p1]);
        maxSum[p] = max( max(maxSum[p1], maxSum[p2]), sufixSum[p1]
  }
}

int rmqSum(int p, int L, int R, int i, int j){
  if ( i > R || j < L) return -1;
  if ( L >= i && R <= j) return sum[p];
  int p1 = rmqSum(left(p), L, (L+R)/2, i, j);
  int p2 = rmqSum(right(p), (L+R)/2+1, R, i, j);
```

```
    if (p1 == -1) return p2;
    if (p2 == -1) return p1;
    return p1+p2;
}

int rmqPrefix(int p, int L, int R, int i, int j){
    if ( i > R || j < L) return -1;
    if ( L >= i && R <= j) return prefixSum[p];
    int p1 = rmqPrefix(left(p), L, (L+R)/2, i, j);
    int p2 = rmqPrefix(right(p), (L+R)/2+1, R, i, j);
    if (p1 == -1) return p2;
    if (p2 == -1) return p1;
    int aux = rmqSum(left(p), L, (L+R)/2, i, j);
    return max(p1, aux + p2);
}

int rmqSufix(int p, int L, int R, int i, int j){
    if ( i > R || j < L) return -1;
    if ( L >= i && R <= j) return sufixSum[p];
    int p1 = rmqSufix(left(p), L, (L+R)/2, i, j);
    int p2 = rmqSufix(right(p), (L+R)/2+1, R, i, j);

    if (p1 == -1) return p2;
    if (p2 == -1) return p1;
    int aux = rmqSum(right(p), (L+R)/2+1, R, i, j);
    return max(p2, aux+p1);
}

int rmq(int p, int L, int R, int i, int j){
    if ( i > R || j < L) return -1;
    if ( L >= i && R <= j) return maxSum[p];
    // int m = (L+R)/2;
```

```
            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;
            int aux1 = rmqPrefix(right(p),(L+R)/2+1,R,i,j);
            int aux2 = rmqSufix(left(p),L,(L+R)/2,i,j);
            return max( max(p1, p2), aux1 + aux2 );
        }

public:
        SegmentTree(const vector <int> &_A){
            A = _A;
            n = (int)A.size();
            sum.assign(4*n,0);
            sufixSum.assign(4*n,0);
            prefixSum.assign(4*n,0);
            maxSum.assign(4*n,0);
            build(1, 0, n-1);
        }

        int rmq(int i, int j){ return rmq(1,0,n-1,i,j); }

};


int main(){

        vector<int> list;
        int n, a,b;

        cin >> n;
```

```
        for(int i=0; i<n; i++){
                cin >> a;
                list.push_back(a);
        }

        SegmentTree st(list);
        cin >> n;
        for(int i=0; i<n; i++){
                cin >> a;
                cin >> b;
                cout << st.rmq(a-1,b-1) << endl;
        }

        return 0;
}
```

10. Dado um vetor de n números inteiros e um inteiro k, desenvolva um algoritmo de complexidade $O(n(\lg n)^2)$ para encontrar o comprimento do menor segmento com mdc igual k.

    Entrada:

    v={6,9,7,10,12,24,36,27}

    k=3

    Saída

    mdc({6,9})=3

    Observe que o mdc(24,36,27) é 3 também, mas {6,9} é o menor segmento.

    Dica: Construa uma árvore de segmento, onde cada nó da árvore tem o mdc do segmento. Depois, faça uma busca binária para encontrar o menor segmento em cada $[i..n]$ para $i = 1$ até n. Observe que se o mdc de um segmento $[i..j]$ for menor que k então nenhum subsegmento pode ter o mdc igual a k.