GEMP - UECE - ICPC Library

Contents

1 Data Structures

1.1 BIT 1D

```
#include <bits/stdc++.h>
using namespace std;

int aux, n, arr[1000], BIT[1000];

// construir uma BIT a partir de um array em O(N)
void build()  {
  for(int i = 1; i <= n; i++) {
    BIT[i] += arr[i];
    if(i + (i & -i) <= n)
      BIT[i + (i & -i)] += BIT[i];
  }
}

// construir o array que gera a BIT a partir de uma BIT em O(N)
void buildArray()  {
  for(int i = n; i >= 1; i--)
    if(i + (i & -i) <= n)
      BIT[i + (i & -i)] -= BIT[i];
}

int sum(int x) {
    int s = 0;
    while(x) s += BIT[x], x -= x&-x;
    return s;
}

void update(int x, int value) {
    while(x <= n) BIT[x] += value, x += x&-x;
}

int main() {
    cin >> n;
    for(int i = 1; i <= n; i++) {
        cin >> aux;
        update(i, aux);
    }
    int a, b;
    cin >> a >> b;
    cout << sum(b)-sum(a-1) << '\n';

    return 0;
}
```

2 Dynamic Programming

2.1 Traveling Salesman Problem

```
#include <bits/stdc++.h>
using namespace std;

int dist[22][22], m;
int memo[20][1 << 20];

int solve(int id, int mask) {
  if(((1 << m) - 1) == mask)
    return dist[id][0];
  if(memo[id][mask] != -1)
    return memo[id][mask];
  int ans = INT_MAX;
  for(int i = 0; i < m; i++)
    if((mask & (1 << i)) == 0)
      ans = min(ans, dist[id][i] + solve(i, mask | (1 << i)));
  return memo[id][mask] = ans;
}

int main() {
  memset(memo, -1, sizeof(memo));
  //inicializa a matriz dist com as distancias
  //de todo mundo pra todo mundo..
  cout << solve(0, 1) << '\n';
  return 0;
}
```

3 Geometry

3.1 Convex Hull

```
#include <bits/stdc++.h>
using namespace std;
#define X first
#define Y second
```

```
typedef pair<int, int> ii;

int cross(ii O, ii A, ii B)
{
    return (((A.X - O.X) * (B.Y - O.Y)) - ((A.Y - O.Y) * (B.X - O.X)))
        ;
}

vector<ii> ConvexHull(vector<ii> P)
{
  if(P.size() <= 1) return P;
    vector<ii> H(2*P.size());
    int k = 0;
    sort(P.begin(), P.end());
    //lower hull
    for(int i = 0; i < P.size(); i++)
    {
        while(k >= 2 and cross(H[k-2], H[k-1], P[i]) < 0) k--;
        H[k++] = P[i];
    }
    //upper hull
    for(int i = P.size()-2, l = k + 1; i >= 0; i--)
    {
        while(k >= l and cross(H[k-2], H[k-1], P[i]) < 0) k--;
        H[k++] = P[i];
    }
    H.resize(k-1);
    return H;
}

int main()
{
    int n, x, y;
    vector<ii> P;

    cin >> n;
    while(n--)
    {
        cin >> x >> y;
        P.push_back({x, y});
    }

    vector<ii> H = ConvexHull(P);

    for(int i = 0; i < H.size(); i++)
        cout << H[i].X << ' ' << H[i].Y << '\n';

  return 0;
}
```

4 Graph

4.1 Bipartite Matching

```
#include <bits/stdc++.h>
using namespace std;

int na, nb, m, tempo = 1;
int b[105];
int cor[105];
vector<int> G[105];

bool kuhn(int u)
{
  if(cor[u] == tempo)
    return 0;
  cor[u] = tempo;
//random_shuffle(G[u].begin(), G[u].end(), [](int x){ return rand() %
    x; });
  for(const int &v : G[u])
    if(!b[v] or kuhn(b[v]))
      return b[v] = u;
  return 0;
}

int main()
{
  //srand(time(NULL));
  cin >> na >> nb >> m;
  while(m--)
  {
    int u, v;
    cin >> u >> v;
    G[u].push_back(v + na);
  }
  tempo = 1;
  int ans = 0;
  for(int i = 1; i <= na; i++)
    ans += kuhn(i), tempo++;
  cout << "MCBM = " << ans << '\n';
  for(int i = nb + 1; i <= na + nb; i++)
    if(b[i])
      cout << b[i] << ' ' << i - na << '\n';

      return 0;
}
```

5 Math

5.1 Baby Step Giant Step

```
// a ^ kcongb mod m

int value[1000008];
int cor[1000008], tempo = 1;

// com vetor o modulo deve ser <= 10^7 fica O(sqrt(m))
inline int discreteLogarithm(int a, int b, int m) {
    tempo++;
  a %= m; b %= m;
    int n = (int)sqrt(m + .0) + 1, an = 1;
  for(int i = 1; i <= n; i++) an = (an * 1LL * a) % m;
    for(int i = 1, cur = an; i <= n; i++) {
      if(cor[cur] < tempo) value[cur] = i, cor[cur] = tempo;
        cur = (cur * 1LL * an) % m;
    }
    for(int j = 0, cur = b; j <= n; j++) {
        if(cor[cur] == tempo) {
            int ans = value[cur] * n - j;
            if(ans < m)
                return ans;
        }
    cur = (cur * 1LL * a) % m;
    }
  return -1;
}

// com mapa o modulo pode ser ateh <= 10^12 fica O(sqrt(m) * log(m))
int discreteLogarithm(int a, int b, int m)
{
  a %= m; b %= m;
    int n = (int)sqrt(m + .0) + 1, an = 1;
  for(int i = 1; i <= n; i++) an = (an * a) % m;
    unordered_map<int, int> value;
    for(int i = 1, cur = an; i <= n; i++) {
      if(!value.count(cur)) value[cur] = i;
        cur = (cur * an) % m;
    }
    for(int j = 0, cur = b; j <= n; j++) {
        if(value[cur]) {
            int ans = value[cur] * n - j;
            if(ans < m)
                return ans;
        }
    cur = (cur * a) % m;
    }
    return -1;
}
```

6 String

6.1 LIS - LDS

```
#include <bits/stdc++.h>
using namespace std;
const int MAX = 1e4;

int n;
int LI[MAX], LD[MAX];
vector<int> arr;

int LIS() {
    for(int i = 0; i < n; i++)
        LI[i] = 1;
    for(int i = n - 1; i >= 0; i--)
        for(int j = 0; j < i; j++)
            if(arr[j] < arr[i])
                LI[j] = max(LI[j], LI[i] + 1);
}

int LDS() {
    reverse(arr.begin(), arr.end());
    for(int i = 0; i < n; i++)
        LD[i] = 1;
    vector<int> pilha;
    for(int i = 0; i < n; i++) {
        int p = (int)(lower_bound(pilha.begin(),
          pilha.end(), arr[i]) - pilha.begin());
        if(p == pilha.size())
            pilha.push_back(arr[i]);
        else
            pilha[p] = arr[i];
        LD[i] = p + 1;
    }

}

int main() {
    cin >> n; arr.resize(n);
    for(int i = 0; i < n; i++) cin >> arr[i];

    LIS();
    LDS();

    for(int i = 0; i < n; i++)
      cout << LI[i] << ' '; puts("");
    for(int i = 0; i < n; i++)
      cout << LD[n - i - 1] << ' '; puts("");

  return 0;
}
```

7 Miscellaneous

7.1 BIT 1D

```
#include <bits/stdc++.h>
using namespace std;

int aux, n, arr[1000], BIT[1000];

// construir uma BIT a partir de um array em O(N)
void build()  {
  for(int i = 1; i <= n; i++) {
    BIT[i] += arr[i];
    if(i + (i & -i) <= n)
      BIT[i + (i & -i)] += BIT[i];
  }
}

// construir o array que gera a BIT a partir de uma BIT em O(N)
void buildArray()  {
  for(int i = n; i >= 1; i--)
    if(i + (i & -i) <= n)
      BIT[i + (i & -i)] -= BIT[i];
}

int sum(int x) {
    int s = 0;
    while(x) s += BIT[x], x -= x&-x;
    return s;
}
```

```
void update(int x, int value) {
    while(x <= n) BIT[x] += value, x += x&-x;
}

int main() {
    cin >> n;
    for(int i = 1; i <= n; i++) {
        cin >> aux;
        update(i, aux);
    }
    int a, b;
    cin >> a >> b;
    cout << sum(b)-sum(a-1) << '\n';

    return 0;
}
```

8 Other