Contents

1 Template

1.1 Eric's Template

```cpp
#include <iostream>
#include <cstdio>
#include <vector>
#include <queue>
#include <string>
#include <cstring>
#include <cmath>
#include <cstdlib>
#include <algorithm>
using namespace std;

#define ll long long
#define C cases
#define PB push_back
#define PP pair<int, int>


int main(void)
{
  ios_base::sync_with_stdio(false);
  cin.tie(0);

  #ifndef file
  freopen("in.in", "r", stdin);
  freopen("out.out", "w", stdout);
  #endif



  return 0;
}
```

2 Graph

2.1 Disjoint Set

```cpp
#define SIZE 10000

int arr[SIZE];

void init(int n) // give a initial length
{
  for(int i=0; i<n; i++)
    arr[i] = -1;
}

int find(int x)
{ // find the father point
  return arr[x] < 0 ? x : (arr[x] = find(arr[x]));
}

void Union(int x, int y)
{
  x = find(x);
  y = find(y);

  if(arr[x] <= arr[y])
  {
    arr[x] += arr[y];
    arr[y] = x;
  }
  else
  {
    arr[y] += arr[x];
    arr[x] = y;
  }
}
```

2.2 Dijkstra

```cpp
//dijkstra
#include <iostream>
#include <cstdio>
#include <queue>
#include <vector>
using namespace std;

#define maxn 51415

struct Edge
{
  int from, to, dist;

  Edge(int _from, int _to, int _dist)
    {
        from = _from;
        to = _to;
        dist = _dist;
    }
};

struct Item
{
    int node;
    int dist;

    Item(int _node, int _dist)
    {
        node = _node;
        dist = _dist;
    }

    bool operator <(const Item& rs) const
    {
        return dist > rs.dist;
    }
};

int main(void)
{
    int n, m;
    while(~scanf("%d %d", &n, &m))
    {
      vector <Edge> edges;
      vector<int> G[maxn];
        priority_queue <Item> dij;
        int visit[maxn] = {-1};

        for(int i = 0; i < m; i++)
        {
            int a, b, c;
            scanf("%d %d %d", &a, &b, &c);
```

```
            edges.push_back(Edge(a, b, c));
            G[a].push_back(i);
        }

        int node = 1;
        dij.push(Item(1, 0));
        Item hold = Item(0, 0);
        while(!dij.empty())
        {
            hold = dij.top();
            dij.pop();

            if(visit[hold.node] == 1)
                continue;

            visit[hold.node] = 1;

            node = hold.node;
            if(node == n)
            {
                break;
            }

            for(int i = 0; i < G[node].size(); i++)
            {
               dij.push(Item(edges[G[node][i]].to, hold
                   .dist+edges[G[node][i]].dist));
            }
        }

        if(node != n) printf("-1\n");
        if(node == n)  printf("%d\n", hold.dist);
    }
    return 0;
}
```

2.3 Kruskal

```
//kruskal algorithm
//minimum spanning tree

#include <iostream>
#include <cstdio>
#include <vector>
#include <algorithm>
#include <cstring>
using namespace std;

#define maxn 10100

struct Edge
{
  int from, to, cost;

  Edge(int _from, int _to, int _cost)
  {
    from = _from;
    to = _to;
    cost = _cost;
  }

  bool operator< (const Edge &r) const
  {
    return cost < r.cost;
  }
};

int graph[maxn];
int parent_arr[maxn];
int n, m;
vector<Edge> edges;

int find(int x)
{
  return parent_arr[x] < 0 ? x : (parent_arr[x] = find(
      parent_arr[x]));
}

void kruskal_algorithm(int vertex, int edge)
{
  int cost = 0;
  memset(parent_arr, -1, sizeof(parent_arr));
  sort(edges.begin(), edges.end());

  for(int i = 0; i < edge; i++)
  {
    Edge tmp = edges[i];

    if(find(tmp.to) == find(tmp.from) && parent_arr[tmp
        .to] != -1)
    {
      //不能形成環的邊
      continue;
    }

    else
    {
      if(parent_arr[tmp.to] == -1)
      {
        parent_arr[find(tmp.from)] += -1;
        parent_arr[tmp.to] = tmp.from;
      }

      else if(parent_arr[tmp.from] == -1)
      {
        parent_arr[find(tmp.from)] -= 1;
        parent_arr[tmp.from] = tmp.to;
      }
      else
      {
        parent_arr[find(tmp.to)] += parent_arr[find(tmp
            .from)];
        parent_arr[find(tmp.from)] = find(tmp.to);
      }
    }
  }
}

int main(void)
{
  while(scanf("%d %d", &n, &m))
  {
    for(int i = 0; i < m; i++)
    {
      int a, b, c;
      scanf("%d %d %d", &a, &b, &c);
      edges.push_back(Edge(a, b, c));

      kruskal_algorithm();

      edges.clear();
    }
  }
  return 0;
}
```

3 Dynamic Programming

3.1 LIS

```
int main(int argc, char const *argv[])
{
    int ls[100050];
    int dp[100050];
    int x;

    cin >> x;

    for(int i = 0; i < x; i++)
    {
```

```
        cin >> ls[i];
    }

    memset(dp, 0, x+5);
    dp[0] = ls[0];
    int top = 0;

    for(int i = 1; i < x; i++)
    {
        int left = 0, right = top;
    bool found = false;

    while(left <= right)
    {
      if(ls[i] > dp[(left+right)/2])  left = (left+
          right)/2+1;
      else if(ls[i] < dp[(left+right)/2])   right = (
          left+right)/2-1;
      else if(ls[i] == dp[(left+right)/2])
      {
        found = true;
        break;
      }
    }
    if(found) continue;
    dp[right+1] = ls[i];

    if(right == top)  top++;
    }

    //test
    for(int i = 0; i < x; i++)
    {
      printf("%d ", dp[i]);
    }
    cout << endl;

    cout << top+1 << endl;
    return 0;
}
```

3.2 LCS

```
#include <iostream>
#include <vector>
using namespace std;

int max(int a, int b)
{
    if(a >= b) return a;
    else return b;
}

int main(int argc, char const *argv[])
{
    int n, m;
    vector <int> lis1;
    vector <int> lis2;
    lis1.push_back(0);
    lis2.push_back(0);

    cin >> n >> m;

    for(int i = 0; i < n; i++)
    {
        int hold;

        cin >> hold;

        lis1.push_back(hold);
    }

    for(int i = 0; i < m; i++)
    {
        int hold;
        cin >> hold;

        lis2.push_back(hold);
    }

    int dp[n+1][m+1];

    for(int i = 0; i <= n; i++)
    {
        dp[i][0] = 0;
    }

    for(int i = 0; i <= m; i++)
    {
        dp[0][i] = 0;
    }

    for(int i = 1; i <= n; i++)
    {
        for(int k = 1; k <= m; k++)
        {
            if(lis1[i] == lis2[k])
            {
                dp[i][k] = dp[i-1][k-1] + 1;
            }

            else
            {
                dp[i][k] = max(dp[i][k-1], dp[i-1][k]);
            }
        }
    }

    cout << dp[n][m] << endl;

    return 0;
}
```

4 Tree

4.1 Segment

```
//segment tree test
//first input the length of the sequence
//input the region left & right and output the min

#include<bits/stdc++.h>

using namespace std;

void build(int input[], int tree[], int left, int right
    , int pos){

  if(left == right){// if find the lowest child return
      the value

    tree[pos] = input[left];
    return ;
  }
  int mid = (left + right) / 2;

  build(input, tree, left, mid, 2*pos);//build left
      child
  build(input, tree, mid+1, right, 2*pos + 1);//build
      right child

  tree[pos] = min(tree[2*pos], tree[2*pos+1]);
}

int range_min(int tree[], int r_left, int r_right, int
    left, int right, int pos){

  if(r_left <= left && r_right >= right)// if total
      overlap
    return tree[pos];
```

```
  if(r_left > right || r_right < left)// none overlap
    return 1000000;
  else{// search the child

    int mid = (left + right) / 2;
    return min(range_min(tree, r_left, r_right, left,
        mid, 2*pos), range_min(tree, r_left, r_right,
        mid+1, right, 2*pos+1));
  }
}

void print(int tree[]){

  for(int i = 1 ; i <= 7 ; i++){

    printf("%d ", tree[i]);
  }
}

int main(void){

  #ifdef DBG
  freopen("1.in", "r", stdin);
  freopen("2.out", "w", stdout);
  #endif

  int len;

  scanf("%d", &len);

  int input[len+1];

  for(int i = 1 ; i <= len ; i++)
    scanf("%d", &input[i]);

  int tree[4*len];
  memset(tree, 0, sizeof(tree));

  build(input, tree, 1, len, 1);
  print(tree);
  putchar('\n');

  int left, right;

  while(~scanf("%d %d", &left, &right))
    printf("min: %d\n", range_min(tree, left, right, 1,
         len, 1));

  return 0;
}
```

5 Number Theory

5.1 Big Integer

```
import java.io.*;
import java.util.Scanner;
import java.math.BigInteger;

public class Main
{
  public static void main(String[] argv)
  {
    Scanner scanner = new Scanner(System.in);

    while(scanner.hasNext())
    {
      String input = scanner.next();
      String input2 = scanner.next();

      BigInteger a = new BigInteger(input);
      BigInteger b = new BigInteger(input2);

      System.out.println("Add: " + a.add(b));
      System.out.println("Sub: " + a.subtract(b));
      System.out.println("Mul: " + a.multiply(b));
      System.out.println("Div: " + a.divide(b));
    }
  }
}
```

5.2 Hanoi

```
#include <iostream>
#include <cstdio>
#include <vector>
#include <algorithm>
using namespace std;

#define ll long long
#define PB push_back
#define PP pair<int, int>
#define A 0
#define B 1
#define C 2

vector<int> towel[3];
int step;
int in_n, in_m;

void print_status(void)
{
  if(step > in_m) return;
  for(int i = A; i <= C; i++)
  {
    printf("%c=>", (char)('A' + i));
    if(towel[i].empty())
    {
      printf("\n");
      continue;
    }
    printf("   ");
    for(int n = 0; n < towel[i].size(); n++)
    {
      if(n < towel[i].size()-1)
        printf("%d ", towel[i][n]);
      else
        printf("%d\n", towel[i][n]);
    }
  }
  printf("\n");
  step++;
}

void move(int n, int from, int tmp, int to)
{
  towel[to].PB(towel[from].back());
  towel[from].pop_back();
  print_status();
}

void hanoi(int n, int from, int tmp, int to)
{
  if(step > in_m) return;
  if(n == 1)
  {
    move(n, from, tmp, to);
    return;
  }
  else
  {
    hanoi(n-1, from, to, tmp);
    move(n, from, tmp, to);
    hanoi(n-1, tmp, from, to);
  }
}

int main(void)
{
  ios_base::sync_with_stdio(false);
```

```
  cin.tie(0);

  #ifndef file
  freopen("in.in", "r", stdin);
  freopen("out.out", "w", stdout);
  #endif

  int cases = 1;

  while(~scanf("%d %d", &in_n, &in_m) && (in_n||in_m))
  {
    step = 0;
    towel[A].clear();
    towel[B].clear();
    towel[C].clear();

    for(int i = in_n; i >= 1; i--)
    {
      towel[A].PB(i);
    }

    printf("Problem #%d\n\n", cases++);
    print_status();
    hanoi(in_n, A, B, C);
  }
  return 0;
}
```

```
{
  ios_base::sync_with_stdio(false);
  cin.tie(0);

  #ifndef file
  freopen("in.in", "r", stdin);
  freopen("out.out", "w", stdout);
  #endif

  ll M, N;

  while(~scanf("%lld %lld", &M, &N))
  {
    if(M == 0 && N == 0)
      break;

    if(N>M)
    {
      ll tmp = M;
      M = N;
      N = tmp;
    }

    printf("%lld\n", M*N*(M+N-2)+(2*N*(N-1)*(3*M-N-1)
        /3));
  }

  return 0;
}
```

5.3 Lower Bound

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <vector>
using namespace std;

#define maxn 10000

int main(void)
{
  int n;
  int find;
  vector <int> arr;

  scanf("%d", &n);

  for(int i = 0; i < n; i++)
  {
    int tmp;
    scanf("%d", &tmp);
    arr.push_back(tmp);
  }

  sort(arr.begin(), arr.end());

  scanf("%d", &find);

  //兩者都在<algorithm> header file

  //找數字是否在array裡面 true = 1, false = 0
  cout << binary_search(arr.begin(), arr.end(), find)
      << endl;

  //找大於或等於那個數的最小'位子'
  printf("%d\n", *lower_bound(arr.begin(), arr.end(),
      find));

  return 0;
}
```

5.4 Queen atack

```
int main(void)
```