Team Notebook

Ari Blondal (SFU)

March 11, 2021

Contents

1 0 - Template

```
#include <bits/stdc++.h>
using namespace std;
// incomplete
#define rep(i, a, b) for(int i = a; i < (b); ++i)
#define all(x) begin(x), end(x)
#define sz(x) (int)(x).size()
typedef long long ll;
typedef pair<int, int> pii;
typedef vector<int> vi;

int main() {
 ios_base::sync_with_stdio(false);
 cin.tie(NULL);
 int t;
 cin >> t;
 while(t--){
  ;
 }
}
```

2 Basics

2.1 Dijkstra

```
/* Taken from cp-algorithms.com */
/* O(m log m) */
const int INF = 1e9;
vi adj[MAXN];

void dij(int s, int n, vi &d, vi &p){
 d.assign(n, INF);
 p.assign(n, -1);

 d[s] = 0;
 priority_queue<pii, vector<pii>, greater<pii>> q;
 q.push({0, s});
    while (!q.empty()) {
        int v = q.top().second;
        int d_v = q.top().first;
        q.pop();
        if (d_v != d[v])
            continue;

        for (auto edge : adj[v]) {
            int to = edge.first;
            int len = edge.second;

            if (d[v] + len < d[to]) {
                d[to] = d[v] + len;
                p[to] = v;
                q.push({d[to], to});
            }
        }
    }
}
```

2.2 Set-Union

```
/* Written by Ari */
/* If joining by size, -UF[root] is size of set */
const int INF = 1e7;
int UF[MAXN] = {0};

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

void init(int n){
 rep(i,0,n) UF[i] = -1;
}

bool join(int a, int b){
 a = find(a), b = find(b);
 if (a==b) return false;
 if (UF[b] < UF[a]) swap(a,b);
 UF[a] += UF[b];
 UF[b] = a;
 return true;
}
```

3 Dynamic Programming

3.1 Longest Increasing Subsequence

```
// From doggy-sweat-cheatsheet by Erfaniaa
void reconstruct_print(int end, int a[], int p[]) {
  int x = end;
  stack<int> s;
  for (; p[x] >= 0; x = p[x]) s.push(a[x]);
  printf("[%d", a[x]);
  for (; !s.empty(); s.pop()) printf(", %d", s.top());
  printf("]\n");
}

int main() {
  int n = 11, A[] = {-7, 10, 9, 2, 3, 8, 8, 1, 2, 3, 4};
  int L[MAX_N], L_id[MAX_N], P[MAX_N];

  int lis = 0, lis_end = 0;
  for (int i = 0; i < n; ++i) {
    int pos = lower_bound(L, L + lis, A[i]) - L;
    L[pos] = A[i];
    L_id[pos] = i;
    P[i] = pos ? L_id[pos - 1] : -1;
    if (pos + 1 > lis) {
      lis = pos + 1;
      lis_end = i;
    }
    printf("LIS ending at A[%d] is of length %d: ", i, pos +
        1);
    reconstruct_print(i, A, P);
    printf("\n");
  }

  printf("Final LIS is of length %d: ", lis);
  reconstruct_print(lis_end, A, P);
  return 0;
}
```

4 Mathematics

4.1 FFT and Multiplication

```
//From "You Know Izad?" team cheatsheet
#define base complex<double>
void fft (vector<base> & a, bool invert){
    if (L(a) == 1) return;
    int n = L(a);
    vector <base> a0(n / 2), a1(n / 2);
    for (int i = 0, j = 0; i < n; i += 2, ++j){
        a0[j] = a[i];
        a1[j] = a[i + 1];
    }
    fft (a0, invert);
    fft (a1, invert);
    double ang = 2 * PI / n * (invert ? -1 : 1);
    base w(1), wn(cos(ang), sin(ang));
    fore(i, 0, n / 2) {
        a[i] = a0[i] + w * a1[i];
        3
        a[i + n / 2] = a0[i] - w * a1[i];
        if (invert)
            a[i] /= 2, a[i + n / 2] /= 2;
        w *= wn;
    }
}
void multiply (const vector<int> &a, const vector<int> & b,
     vector<int> &res){
    vector <base> fa(all(a)), fb(all(b));
    size_t n = 1;
    while (n < max(L(a), (L(b)))) n <<= 1;
    n <<= 1;
    fa.resize(n), fb.resize(n);
    fft(fa, false), fft(fb, false);
    fore(i, 0, n)
    fa[i] *= fb[i];
    fft (fa, true);
    res.resize (n);
    fore(i, 0, n)
    res[i] = int (fa[i].real() + 0.5);
}
```