REPORT

Competitive Programming

IET SMP

Shanthanu S Rai

7/17/2018

This report contains an overview of the mentorship. It begins with problem definition and an overview of the sessions conducted in the SMP. It also contains links to all assignment problems and contest problems I have solved. It ends with my future plans after this mentorship.

Acknowledgements

I would like to thank both the mentors Naveen and Abhishek who did a really good job in introducing me to new topics in competitive programming and were always there to clear my doubts. I feel I will be a better competitive programmer because of this mentorship.

I would also like to thank all the students in this mentorship who actively participated in contests and gave good competition, thus helping me to improve my skills and sharpen my knowledge in competitive programming.

Table of Contents

Problem Definition

Competitive programming is solving *well-defined problems* by writing *computer programs* under *specified limits*.

Based on the above definition, competitive programming has three aspects:

- **Well-defined problems**. You are presented with one or more problems. The problem statement contains variables, and you have to be able to answer the problem if given any possible combination of values of the variables. The problem will be well-defined: you will be informed the exact constraints of all variables, any necessary assumptions, etc.
- **Computer programs**. You write computer programs that solve the problems. Note that the "computer program" here is a very simple command-line program; no fancy GUI or web app etc. The command-line program reads the values of the variables from the standard input, and must write the answer to the standard output.
- **Specified limits**. Your program must run and produce the answer within a specified time and memory limit. Also, you must write the programs in a specified set of allowed programming languages.

Overview

In this mentorship, we had sessions for the following topics:

- Introduction to competitive programming and STL in C++
- Game theory
- Maths for CP and linear data structure
- Greedy paradigm and bit manipulation
- Trees and Graphs(BFS and DFS)
- Dynamic Programming
- More graph algorithms(MST, Dijkstra's, Bellman Ford)
- Segment Trees for range queries
- Square root decomposition

We also had assignments for after each session. Two contests were conducted during the mentorship.

The mentorship ended with a final contest.

I learnt a lot of new concepts that I was previously not aware of. Segment Trees and Square Root Decomposition were new topics for me which I wasn't aware of. The sessions too were followed by interesting assignment problems which enabled me to creatively apply the concepts used in the respective session. The contests were a compilation of problems of varying difficulty. Overall, it was an awesome experience.

Code

Assignments:

https://drive.google.com/open?id=1di2ofjFq5cjZv_M6Dn7xriNCSMX9ik0l

Contests:

Contest 1:

1. **Pop Count**

```
#include<iostream>
using namespace std;
typedef long long ll;
int main() {
	int n, q;
  cin>>n>>q;
	ll pre[n+1] = {0};
	for(int i = 1; i <= n; i++) {
		ll temp;
		cin>>temp;
		pre[i] = temp+pre[i-1];
	}
	while(q--) {
		ll l,r;
		cin>>l>>r;
		ll sum = (pre[r] - pre[--l])%4294967296;
		int count = 0;
		for(int i = 0; i <= 31; i++) {
			ll set_bit = 1<<i;
			if((sum & set_bit) != 0) {
				count++;
			}
		}
		cout<<count<<endl;
	}
	return 0;
}
```

2. **Lets Play Odd Even**

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


int main() {
```

```
    int t;
    cin>>t;
    while(t--) {
      int n, p, q, r, size_o = 0, size_e = 0, size_oe = 0;
      cin>>n>>p>>q>>r;
      while(n--) {
        int a;
        cin>>a;
        if(a%2 == 1) {
          size_o++;
        }
        else if(a%4 != 0) {
          size_oe++;
        }
        else {
          if(q>r) {
            size_e++;
          }
          else {
            size_oe++;
          }
        }
      }
      cout<<(p*size_o+q*size_e+r*size_oe)<<endl;
    }
    return 0;
}
```

3. Van Helsing snares Dracula

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


int main() {
    long long n;
    cin>>n;
    bool flag = true;
    int count = 1;
    while(n != 1) {
      if(n%4 == 0) {
        n/=4;
      }
      else if(n%4 == 1) {
        n/=4;
        count++;
      }
```

```
        else {
            flag = false;
            break;
        }
    }
    if(flag) {
        cout<<count;
    }
    else {
        cout<<-1;
    }
    return 0;
}
```

4. Shipment of Toys

```
#include <bits/stdc++.h>

using namespace std;
int n;

int toys(vector <int> w) {
    sort(w.begin(), w.end());
    int count = 0, prev = w[0];
    for(int i = 0; i != n; i++) {
        if(w[i] - prev > 4) {
            count++;
            prev = w[i];
        }
    }
    count++;
    return count;
}

int main() {
    cin >> n;
    vector<int> w(n);
    for(int w_i = 0; w_i < n; w_i++){
      cin >> w[w_i];
    }
    int result = toys(w);
    cout << result << endl;
    return 0;
}
```

5. Game of Numbers

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

```
int main() {
   int q;
   cin>>q;
   while(q--) {
      int n;
      cin>>n;
      long double s = sqrt(n);
      if(s-floor(s) == 0) {
         cout<<'B';
      }
      else {
         cout<<'A';
      }
      cout<<endl;
   }
   return 0;
}
```

Contest 2:

1. Munni vs Bunny

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


int main() {
   int t;
   cin>>t;
   while(t--) {
      int n;
      cin>>n;
      if(n%4 != 0) {
         cout<<"Munni"<<endl;
      }
      else {
         cout<<"Bunny"<<endl;
      }
   }
   return 0;
}
```

2. Trip to Bahamas

```
#include <cstdio>
```

```
#include <vector>
#include <iostream>
#include <algorithm>
#include <queue>
#define pb push_back
using namespace std;

vector<vector<int>> adj;
vector<bool> vis;

int bfs(int i) {
   int count = 1;
   queue<int> Q;
   Q.push(i);
   vis[i] = true;
   while(!Q.empty()) {
      int v = Q.front();
      Q.pop();
      for(auto u:adj[v]) {
         if(!vis[u]) {
            count++;
            Q.push(u);
            vis[u] = true;
         }
      }
   }
   return count;
}
```

```
int main() {
   int n,m;
   cin>>n>>m;
   adj.assign(n, vector<int>());
   vis.assign(n, false);
   for(int i = 0; i < m; i++) {
      int u,v;
      cin>>u>>v;
      u--;v--;
      adj[u].pb(v);
      adj[v].pb(u);
   }
   int s = 0;
   for(int i = 0; i < n; i++) {
      if(!vis[i]) {
         int count = bfs(i);
         if(count > s) {
            s = count;
         }
      }
   }
   cout<<s;
   return 0;
}
```

3. Am I Special?

```
#include <bits/stdc++.h>

using namespace std;
#define pb push_back
typedef long long ll;

class dus {
```

```
        private:
                vector<int> parent, size;
                int n; //number of elements
        public:
                dus(int n); //input number of elements
                void make_set(int v);
                void union_sets(int a, int b);
                int find_set(int v);
};

dus::dus(int n) {
        parent.assign(n,0);
        size.assign(n,0);
        for(int i = 0; i < n; i++) {
                make_set(i);
        }
}

void dus::make_set(int v) {
   parent[v] = v;
   size[v] = 1;
}

void dus::union_sets(int a, int b) {
   a = find_set(a);
   b = find_set(b);
   if (a != b) {
      if (size[a] < size[b])
         swap(a, b);
      parent[b] = a;
      size[a] += size[b];
   }
}

int dus::find_set(int v) {
   if (v == parent[v])
      return v;
   return parent[v] = find_set(parent[v]);
}

int main()
{
   int n,m;
   cin>>n>>m;
   vector<pair<ll, pair<int, int> > > w;
   vector<vector<int> > adj(n);
   for(int i = 0; i < m; i++) {
      ll u,v,temp;
      cin>>u>>v>>temp;
```

```
        u--;v--;
        pair<int, pair<int, int> > p = {temp, {u,v}};
        w.pb(p);
        adj[u].pb(v);
        adj[v].pb(u);
      }
      sort(w.begin(), w.end());
      ll count = 0, sum = 0;
      dus a(n);
      for(auto p : w) {
        if(count == n-1) {
          break;
        }
        ll weight = p.first;
        int u = p.second.first, v = p.second.second;
        //cout<<u<<" "<<v<<" "<<weight<<endl;
        if(a.find_set(u) != a.find_set(v)) {
          sum += weight;
          //cout<<u<<" "<<v<<" "<<weight<<endl;
          a.union_sets(u,v);
          count++;
        }
      }
      cout<<sum;
      return 0;
    }
```

4. Diaz in Vice City

```
// Program to find Dijkstra's shortest path using
// priority_queue in STL
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
# define INF 1000000000000000

// iPair ==>  lleger Pair
typedef pair<ll, ll> iPair;

// This class represents a directed graph using
// adjacency list representation
class Graph
{
  ll V;   // No. of vertices

  // In a weighted graph, we need to store vertex
  // and weight pair for every edge
  list< pair<ll, ll> > *adj;

public:
  Graph(ll V);  // Constructor
```

```
  // function to add an edge to graph
  void addEdge(ll u, ll v, ll w);

  // prlls shortest path from s
  void shortestPath(ll s);
};

// Allocates memory for adjacency list
Graph::Graph(ll V)
{
  this->V = V;
  adj = new list<iPair> [V];
}

void Graph::addEdge(ll u, ll v, ll w)
{
  adj[u].push_back(make_pair(v, w));
  adj[v].push_back(make_pair(u, w));
}

// Prlls shortest paths from src to all other vertices
void Graph::shortestPath(ll src)
{
  // Create a priority queue to store vertices that
  // are being preprocessed. This is weird syntax in C++.
  // Refer below link for details of this syntax
  // http://geeksquiz.com/implement-min-heap-using-stl/
  priority_queue< iPair, vector <iPair> , greater<iPair> > pq;

  // Create a vector for distances and initialize all
  // distances as infinite (INF)
  vector<ll> dist(V, INF);

  // Insert source itself in priority queue and initialize
  // its distance as 0.
  pq.push(make_pair(0, src));
  dist[src] = 0;

  /* Looping till priority queue becomes empty (or all
    distances are not finalized) */
  while (!pq.empty())
  {
    // The first vertex in pair is the minimum distance
    // vertex, extract it from priority queue.
    // vertex label is stored in second of pair (it
    // has to be done this way to keep the vertices
    // sorted distance (distance must be first item
    // in pair)
```

```
        ll u = pq.top().second;
        pq.pop();

        // 'i' is used to get all adjacent vertices of a vertex
        list< pair<ll, ll> >::iterator i;
        for (i = adj[u].begin(); i != adj[u].end(); ++i)
        {
          // Get vertex label and weight of current adjacent
          // of u.
          ll v = (*i).first;
          ll weight = (*i).second;

          //  If there is shorted path to v through u.
          if (dist[v] > dist[u] + max((ll)0, weight - dist[u]))
          {
            // Updating distance of v
            dist[v] = dist[u] + max((ll)0, weight - dist[u]);
            pq.push(make_pair(dist[v], v));
          }
        }
      }

      // Prll shortest distances stored in dist[]
      if(dist[V-1] == INF) {
        cout<<-1;
      }
      else {
        cout<<dist[V-1];
      }
    }
    int main() {
      ll n,m;
            cin>>n>>m;
      Graph g(n);
            for(ll i = 0; i < m; i++) {
                  ll u,v,w;
                  cin>>u>>v>>w;
                  u--;v--;
                  //undirected
                  g.addEdge(u, v, w);
            }
      g.shortestPath(0);
      return 0;
    }
```

5. XOR on Arrays

```
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
```

```
#include <algorithm>
using namespace std;
typedef long long ll;

ll a[2*100000+1] = {0};

inline int pow2roundup (int x)
{
   if (x < 0)
      return 0;
   --x;
   x |= x >> 1;
   x |= x >> 2;
   x |= x >> 4;
   x |= x >> 8;
   x |= x >> 16;
   return x+1;
}


void find_xor(vector<ll>& v, int x1, int y1, int x2, int y2) {
   //cout<<x1<<" "<<y1<<" "<<x2<<" "<<y2<<endl;
   if(x2 == x1+1) {//base case
      v[x1] ^= a[y1];
      return;
   }
   int add = (x2-x1)/2;
   find_xor(v,x1+add,y1,x2,y2-add);
   find_xor(v,x1,y1+add,x2-add,y2);
   for(int i = x1; i < x1+add; i++) {
      v[i] ^= v[i+add];
   }
}

int main() {
   unsigned int n,q;
   cin>>n>>q;
   for(int i = 0; i < n; i++) cin>>a[i];
   n = pow2roundup(n);
   vector<ll> v(n,0);
   find_xor(v,0,0,n,n);
            reverse(v.begin(), v.end());
   reverse(v.begin()+1, v.end());
   /*for(int i = 0; i < n; i++) {
      cout<<v[i]<<endl;
   }*/
   while(q--) {
                        ll d;
      cin>>d;
```

```
    d = d%n;
    cout<<v[d]<<endl;
  }
  return 0;
}
```

Final Contest:

1. Lesser Primes

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

int soe(int n)
{
  int count = 0;
  bool prime[n+1];
  memset(prime, true, sizeof(prime));

  for (int p=2; p*p<=n; p++)
  {
    if (prime[p] == true)
    {
      // Update all multiples of p
      for (int i=p*2; i<=n; i += p)
        prime[i] = false;
    }
  }
  for (int p=2; p<=n; p++)
    if (prime[p])
      count++;
  return count;
}

int main() {
  int n;
  cin>>n;
  cout<<soe(n-1);
  return 0;
}
```

2. Bunny loves U

```
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;
typedef long long ll;

ll f(ll n) {
```

```
    if(n == 0) return 0;
    if(n == 1) return 0;
    return (n)*(n-1)/2;
}

int main() {
    string s;
    cin>>s;
    ll ans = f(s.size()+1), prev = -1;
    for(int i = 0; i < s.size(); i++) {
        if(s[i] == 'u') {
            ans -= f(i-prev);
            prev = i;
        }
    }
    ans -= f(s.size()-prev);
    cout<<ans;
    return 0;
}
```

3. Play to Win

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


int main() {
    int t, n, temp;
    cin>>t;
    while(t--) {
        int count[3] = {0};
        cin>>n;
        for(int i = 0; i < n; i++) {
            cin>>temp;
            count[temp%3]++;
        }
        if(count[1]%2 == 0 && count[2]%2 == 0) cout<<"Koca"<<endl;
        else cout<<"Balsa"<<endl;
    }
    return 0;
}
```

4. Make the arrays same

```
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
```

```
using namespace std;
typedef long long ll;

int main() {
   int n;
   cin>>n;
   ll a[n], b[n];
   for(int i = 0; i < n; i++) cin>>a[i];
   for(int i = 0; i < n; i++) cin>>b[i];
   ll move = 0;
   for(int i = 0; i < n; i++) {
      if(a[i] >= b[i]) move += a[i]-b[i];
      else {
         int max = ((b[i]%2==0)?b[i]/2-1:b[i]/2);
         move++;
         if(a[i]>max) move+=(a[i]-max);
      }
   }
   cout<<move;
   return 0;
}
```

5. Choice of Seats

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


int main() {
   string s;
   cin>>s;
   if(s[s.size()-1] == 'E') {
      cout<<(s.size()-1);
      return 0;
   }
   if(s[0] == 'E') {
      cout<<0;
      return 0;
   }
   int prev = -1, ans = -1;
   for(int i = 0; i < s.size(); i++) {
      if(s[i] == 'O') {
         int temp = i-prev-1;
         int p = prev;
         prev = i;
         if(temp == 0) continue;
         if(temp == 1) ans = p+1;
```

```
            if(temp == 2) ans = i-1;
            if(temp>2) {
               ans = p + 2;
               break;
            }
          }
        }
        cout<<ans;
        return 0;
}
```

6. Cyclic array problem

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

struct SegmentTreeNode {
 int maxi, mini;

 void assignLeaf(int value) {
  maxi = mini = value;
 }

 void merge(SegmentTreeNode& left, SegmentTreeNode& right) {
  mini = min(left.mini, right.mini);
  maxi = max(left.maxi, right.maxi);
 }

 int getValue() {
  return (maxi-mini);
 }
};

// T is the type of input array elements
// V is the type of required aggregate statistic
template<class T, class V>
class SegmentTree {
 SegmentTreeNode* nodes;
 int N;

public:
 SegmentTree(T arr[], int N) {
  this->N = N;
  nodes = new SegmentTreeNode[getSegmentTreeSize(N)];
  buildTree(arr, 1, 0, N-1);
 }
```

```
~SegmentTree() {
 delete[] nodes;
}

// V is the type of the required aggregate statistic
V getValue(int lo, int hi) {
 SegmentTreeNode result = getValue(1, 0, N-1, lo, hi);
 return result.getValue();
}

// We want to update the value associated with index in the input array
void update(int index, T value) {
 update(1, 0, N-1, index, value);
}

private:
void buildTree(T arr[], int stIndex, int lo, int hi) {
 if (lo == hi) {
  nodes[stIndex].assignLeaf(arr[lo]);
  return;
 }

 int left = 2 * stIndex, right = left + 1, mid = (lo + hi) / 2;
 buildTree(arr, left, lo, mid);
 buildTree(arr, right, mid + 1, hi);
 nodes[stIndex].merge(nodes[left], nodes[right]);
}

SegmentTreeNode getValue(int stIndex, int left, int right, int lo, int hi) {
 if (left == lo && right == hi)
  return nodes[stIndex];

 int mid = (left + right) / 2;
 if (lo > mid)
  return getValue(2*stIndex+1, mid+1, right, lo, hi);
 if (hi <= mid)
  return getValue(2*stIndex, left, mid, lo, hi);

 SegmentTreeNode leftResult = getValue(2*stIndex, left, mid, lo, mid);
 SegmentTreeNode rightResult = getValue(2*stIndex+1, mid+1, right, mid+1, hi);
 SegmentTreeNode result;
 result.merge(leftResult, rightResult);
 return result;
}

int getSegmentTreeSize(int N) {
 int size = 1;
 for (; size < N; size <<= 1);
 return size << 1;
```

```
 }

 void update(int stIndex, int lo, int hi, int index, T value) {
  if (lo == hi) {
  nodes[stIndex].assignLeaf(value);
  return;
  }

  int left = 2 * stIndex, right = left + 1, mid = (lo + hi) / 2;
  if (index <= mid)
   update(left, lo, mid, index, value);
  else
   update(right, mid+1, hi, index, value);

  nodes[stIndex].merge(nodes[left], nodes[right]);
 }
};

int main() {
   int n, q, l, r;
   cin>>n;
   int a[2*n];
   for(int i = 0; i < n; i++) cin>>a[i];
   for(int i = 0; i < n; i++) a[n+i] = a[i];
   SegmentTree<int, int> st(a, 2*n);
   cin>>q;
   while(q--) {
      cin>>l>>r;
      l--;r--;
      if(l>r) r += n;
      cout<<st.getValue(l,r)<<endl;
   }
   return 0;
}
```

7. Good Subsequence

```
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;
typedef long long ll;
const int p = 1e9+7;

ll power(ll x, ll y, ll p)
{
   ll res = 1;     // Initialize result

   x = x % p;  // Update x if it is more than or
```

```
            // equal to p

    while (y > 0)
    {
        // If y is odd, multiply x with result
        if (y & 1)
            res = (res*x) % p;

        // y must be even now
        y = y>>1; // y = y/2
        x = (x*x) % p;
    }
    return res;
}

// Returns n^(-1) mod p
ll modInverse(ll n, ll p)
{
    return power(n, p-2, p);
}

ll fac[500001];
// Returns nCr % p using Fermat's little
// theorem.
ll nCr(ll n, ll r, ll p)
{
   // Base case
   if (r==0)
      return 1;

    // Fill factorial array so that we
    // can find all factorial of r, n
    // and n-r

    return (fac[n]* modInverse(fac[r], p) % p *
            modInverse(fac[n-r], p) % p) % p;
}

int main() {
    fac[0] = 1;
    for (ll i=1 ; i<=500000; i++)
        fac[i] = fac[i-1]*i%p;
    ll q;
    cin>>q;
    while(q--) {
        string s;
        cin>>s;
        ll count[4] = {0};
        for(auto c:s) count[c-'a']++;
```

```
      cout<<((nCr(count[0]+count[1], count[0], p)*nCr(count[2]+count[3], count[2], p) -
1)%p)<<endl;
   }
   return 0;
}
```

Future Work

Having a solid foundation in competitive programming, I plan to further enhance my knowledge and sharpen my skills. I wish to develop a good intuition in algorithms and data structures and use my knowledge to implement efficient algorithms and come up with innovative solutions for various problems.

I will also start preparing ACM ICPC contest and hope to do well in the contest.