KUET_ThunderBolt Team Notebook

June 1, 2022

Muhiminul Islam Osim, Toufiq Imam Nuhash, Sharif Minhazul Islam

Contents

1 Data Structure 1
1.1 template [30 lines] 1

2 Dynamic Programming 1
2.1 template [30 lines] 1

3 Geometry 1
3.1 template [30 lines] 1

4 Graph 2
4.1 template [30 lines] 2

5 Math 2
5.1 template [30 lines] 2

6 Misc 2
6.1 template [30 lines] 2

7 String 3
7.1 Aho-Corasick [124 lines] 3
7.2 Double Hasing [50 lines] 4
7.3 KMP [23 lines] 4
7.4 Palindromic Tree [30 lines] 4
7.5 Suffix Array [78 lines] 5
7.6 Trie [28 lines] 5
7.7 Z-Algorithm [19 lines] 5

1 Data Structure

1.1 template [30 lines]

```
#include <bits/stdc++.h>
using namespace std;
template <typename A, typename B> ostream
    &operator<<(ostream &os, const pair<A, B> &p) {
    return os << '(' << p.first << ", " << p.second <<
    ')'; }
template <typename T_container, typename T = typename
    enable_if<!is_same<T_container, string>::value,
    typename T_container::value_type>::type> ostream
    &operator<<(ostream &os, const T_container &v) {
    os << '{'; string sep; for (const T &x : v) os <<
    sep << x, sep = ", "; return os << '}'; }
void dbg_out() { cerr << endl; }
template <typename Head, typename... Tail> void
    dbg_out(Head H, Tail... T) { cerr << " " << H;
    dbg_out(T...); }
#ifdef SMIE
#define debug(args...) cerr << "(" << #args << "):",
    dbg_out(args)
#else
#define debug(args...)
#endif

mt19937
    rng(chrono::steady_clock::now().time_since_epoch().count());

//uniform_int_distribution<int>(0, i)(rng)

void solve()
{
}

int main()
{
    ios_base::sync_with_stdio(false); //DON'T mix C
    and C++ I/O
    cin.tie(NULL);                    //DON'T use for
    interactive problem
    int tests = 1;
    cin >> tests;
    while (tests--) {
        solve();
    }
}
```

2 Dynamic Programming

2.1 template [30 lines]

```
#include <bits/stdc++.h>
using namespace std;
template <typename A, typename B> ostream
    &operator<<(ostream &os, const pair<A, B> &p) {
    return os << '(' << p.first << ", " << p.second <<
    ')'; }
template <typename T_container, typename T = typename
    enable_if<!is_same<T_container, string>::value,
    typename T_container::value_type>::type> ostream
    &operator<<(ostream &os, const T_container &v) {
    os << '{'; string sep; for (const T &x : v) os <<
    sep << x, sep = ", "; return os << '}'; }
void dbg_out() { cerr << endl; }
template <typename Head, typename... Tail> void
    dbg_out(Head H, Tail... T) { cerr << " " << H;
    dbg_out(T...); }
#ifdef SMIE
#define debug(args...) cerr << "(" << #args << "):",
    dbg_out(args)
#else
#define debug(args...)
#endif

mt19937
    rng(chrono::steady_clock::now().time_since_epoch().c

//uniform_int_distribution<int>(0, i)(rng)

void solve()
{
}

int main()
{
    ios_base::sync_with_stdio(false); //DON'T mix C
    and C++ I/O
    cin.tie(NULL);                    //DON'T use for
    interactive problem
    int tests = 1;
    cin >> tests;
    while (tests--) {
        solve();
    }
}
```

3 Geometry

3.1 template [30 lines]

```
#include <bits/stdc++.h>
using namespace std;
template <typename A, typename B> ostream
    &operator<<(ostream &os, const pair<A, B> &p) {
    return os << '(' << p.first << ", " << p.second <<
    ')'; }
template <typename T_container, typename T = typename
    enable_if<!is_same<T_container, string>::value,
    typename T_container::value_type>::type> ostream
    &operator<<(ostream &os, const T_container &v) {
    os << '{'; string sep; for (const T &x : v) os <<
    sep << x, sep = ", "; return os << '}'; }
void dbg_out() { cerr << endl; }
```

```
template <typename Head, typename... Tail> void
    dbg_out(Head H, Tail... T) { cerr << " " << H;
    dbg_out(T...); }
#ifdef SMIE
#define debug(args...) cerr << "(" << #args << "):",
    dbg_out(args)
#else
#define debug(args...)
#endif

mt19937
    rng(chrono::steady_clock::now().time_since_epoch().count());

//uniform_int_distribution<int>(0, i)(rng)

void solve()
{
}

int main()
{
    ios_base::sync_with_stdio(false); //DON'T mix C
    and C++ I/O
    cin.tie(NULL);                    //DON'T use for
    interactive problem
    int tests = 1;
    cin >> tests;
    while (tests--) {
        solve();
    }
}
```

4 Graph

4.1 template [30 lines]

```
#include <bits/stdc++.h>
using namespace std;
template <typename A, typename B> ostream
    &operator<<(ostream &os, const pair<A, B> &p) {
    return os << '(' << p.first << ", " << p.second <<
    ')'; }
template <typename T_container, typename T = typename
    enable_if<!is_same<T_container, string>::value,
    typename T_container::value_type>::type> ostream
    &operator<<(ostream &os, const T_container &v) {
    os << '{'; string sep; for (const T &x : v) os <<
    sep << x, sep = ", "; return os << '}'; }
void dbg_out() { cerr << endl; }
template <typename Head, typename... Tail> void
    dbg_out(Head H, Tail... T) { cerr << " " << H;
    dbg_out(T...); }
#ifdef SMIE
#define debug(args...) cerr << "(" << #args << "):",
    dbg_out(args)
#else
#define debug(args...)
#endif

mt19937
    rng(chrono::steady_clock::now().time_since_epoch().count());

//uniform_int_distribution<int>(0, i)(rng)

void solve()
{
}

int main()
{
    ios_base::sync_with_stdio(false); //DON'T mix C
    and C++ I/O
    cin.tie(NULL);                    //DON'T use for
    interactive problem
    int tests = 1;
    cin >> tests;
    while (tests--) {
        solve();
    }
}
```

5 Math

5.1 template [30 lines]

```
#include <bits/stdc++.h>
using namespace std;
template <typename A, typename B> ostream
    &operator<<(ostream &os, const pair<A, B> &p) {
    return os << '(' << p.first << ", " << p.second <<
    ')'; }
template <typename T_container, typename T = typename
    enable_if<!is_same<T_container, string>::value,
    typename T_container::value_type>::type> ostream
    &operator<<(ostream &os, const T_container &v) {
    os << '{'; string sep; for (const T &x : v) os <<
    sep << x, sep = ", "; return os << '}'; }
void dbg_out() { cerr << endl; }
template <typename Head, typename... Tail> void
    dbg_out(Head H, Tail... T) { cerr << " " << H;
    dbg_out(T...); }
#ifdef SMIE
#define debug(args...) cerr << "(" << #args << "):",
    dbg_out(args)
#else
#define debug(args...)
#endif

mt19937
    rng(chrono::steady_clock::now().time_since_epoch().c

//uniform_int_distribution<int>(0, i)(rng)

void solve()
{
}

int main()
{
    ios_base::sync_with_stdio(false); //DON'T mix C
    and C++ I/O
    cin.tie(NULL);                    //DON'T use for
    interactive problem
    int tests = 1;
    cin >> tests;
    while (tests--) {
        solve();
    }
}
```

6 Misc

6.1 template [30 lines]

```
#include <bits/stdc++.h>
using namespace std;
template <typename A, typename B> ostream
    &operator<<(ostream &os, const pair<A, B> &p) {
    return os << '(' << p.first << ", " << p.second <<
    ')'; }
```

```cpp
template <typename T_container, typename T = typename
    enable_if<!is_same<T_container, string>::value,
    typename T_container::value_type>::type> ostream
    &operator<<(ostream &os, const T_container &v) {
    os << '{'; string sep; for (const T &x : v) os <<
    sep << x, sep = ", "; return os << '}'; }
void dbg_out() { cerr << endl; }
template <typename Head, typename... Tail> void
    dbg_out(Head H, Tail... T) { cerr << " " << H;
    dbg_out(T...); }
#ifdef SMIE
#define debug(args...) cerr << "(" << #args << "):",
    dbg_out(args)
#else
#define debug(args...)
#endif

mt19937
    rng(chrono::steady_clock::now().time_since_epoch().count());

//uniform_int_distribution<int>(0, i)(rng)

void solve()
{
}

int main()
{
    ios_base::sync_with_stdio(false); //DON'T mix C
    and C++ I/O
    cin.tie(NULL);                    //DON'T use for
    interactive problem
    int tests = 1;
    cin >> tests;
    while (tests--) {
        solve();
    }
}
```

7 String

7.1 Aho-Corasick [124 lines]

```cpp
const int NODE=3000500;///Maximum Nodes
const int LGN=30;      ///Maximum Number of Tries
const int MXCHR=53;    ///Maximum Characters
const int MXP=5005;    ///
struct node {
  int val;
  int child[MXCHR];
  vector<int>graph;
  void clear(){
    CLR(child,0);
    val=0;
    graph.clear();
  }
}Trie[NODE+10];
int maxNodeId,fail[NODE+10],par[NODE+10];
int nodeSt[NODE+10],nodeEd[NODE+10];
vlong csum[NODE+10],pLoc[MXP];
void resetTrie(){
  maxNodeId=0;
}
int getNode(){
  int curNodeId=++maxNodeId;
  Trie[curNodeId].clear();
  return curNodeId;
}
inline void upd(vlong pos){
  csum[pos]++;
}
inline vlong qry(vlong pos){
  vlong res=csum[pos];
  return res;
}
struct AhoCorasick {
  int root,size,euler;
  void clear(){
    root=getNode();
    size=euler=0;
  }
  inline int getname(char ch){
    if(ch=='-')return 52;
    else if(ch>='A' && ch<='Z')return 26+(ch-'A');
    else return(ch-'a');
  }
  void addToTrie(string &s,int id){
  //Add string s to the Trie in general way
    int len=SZ(s),cur=root;
    FOR(i,0,len-1){
      int c=getname(s[i]);
      if(Trie[cur].child[c]==0){
        int curNodeId=getNode();
        Trie[curNodeId].val=c;
        Trie[cur].child[c]=curNodeId;
      }
      cur=Trie[cur].child[c];
    }
    pLoc[id]=cur;
    size++;
  }
  void calcFailFunction(){
    queue<int>Q;
    Q.push(root);
    while(!Q.empty()){
      int s=Q.front();
      Q.pop();
    //Add all the children to the queue:
      FOR(i,0,MXCHR-1){
        int t=Trie[s].child[i];
        if(t!=0){
          Q.push(t);
          par[t]=s;
        }
      }
      if(s==root){/*Handle special case when s is
    root*/
        fail[s]=par[s]=root;
        continue;
      }
   //Find fall back of s:
      int p=par[s],f=fail[p];;
      int val=Trie[s].val;
   /*Fall back till you found a node who has got val as
    a child*/
      while(f!=root && Trie[f].child[val]==0){
        f=fail[f];
      }
      fail[s]=(Trie[f].child[val]==0)? root :
    Trie[f].child[val];
   //Self fall back not allowed
      if(s==fail[s]){
        fail[s]=root;
      }
      Trie[fail[s]].graph.push_back(s);
    }
  }
  void dfs(int pos){
    ++euler;
    nodeSt[pos]=euler;
    for(auto x: Trie[pos].graph){
      dfs(x);
    }
    nodeEd[pos]=euler;
  }
 //Returns the next state
  int goTo(int state,int c){
    if(Trie[state].child[c]!=0){/*No need to fall
    back*/
      return Trie[state].child[c];
    }
  //Fall back now:
    int f=fail[state];
    while(f!=root && Trie[f].child[c]==0){
      f=fail[f];
    }
    int res=(Trie[f].child[c]==0)?
    root:Trie[f].child[c];
    return res;
  }
 /*Iterate through the whole text and find all the
    matchings*/
  void findmatching(string &s){
    int cur=root,idx=0;
    int len=SZ(s);
    while(idx<len){
      int c=getname(s[idx]);
      cur=goTo(cur,c);
      upd(nodeSt[cur]);
      idx++;
    }
  }
}acorasick;
```

7.2 Double Hasing [50 lines]

```
struct SimpleHash {
    int len;
    long long base, mod;
    vector<int> P, H, R;
    SimpleHash() {}
    SimpleHash(const char* str, long long b, long long
    m) {
        base = b, mod = m, len = strlen(str);
        P.resize(len + 4, 1), H.resize(len + 3, 0),
    R.resize(len + 3, 0);
        for (int i = 1; i <= len + 3; i++)
            P[i] = (P[i - 1] * base) % mod;
        for (int i = 1; i <= len; i++)
            H[i] = (H[i - 1] * base + str[i - 1] +
    1007) % mod;
        for (int i = len; i >= 1; i--)
            R[i] = (R[i + 1] * base + str[i - 1] +
    1007) % mod;
    }
    inline int range_hash(int l, int r) {
        int hashval = H[r + 1] - ((long long)P[r - l
    + 1] * H[l] % mod);
        return (hashval < 0 ? hashval + mod :
    hashval);
    }
    inline int reverse_hash(int l, int r) {
        int hashval = R[l + 1] - ((long long)P[r - l
    + 1] * R[r + 2] % mod);
        return (hashval < 0 ? hashval + mod :
    hashval);
    }
};
struct DoubleHash {
    SimpleHash sh1, sh2;
    DoubleHash() {}
    DoubleHash(const char* str) {
        sh1 = SimpleHash(str, 1949313259, 2091573227);
        sh2 = SimpleHash(str, 1997293877, 2117566807);
    }
    long long concate(DoubleHash& B , int l1 , int r1
    , int l2 , int r2) {
        int len1 = r1 - l1+1 , len2 = r2 - l2+1;
        long long x1 = sh1.range_hash(l1, r1) ,
        x2 = B.sh1.range_hash(l2, r2);
        x1 = (x1 * B.sh1.P[len2]) % 2091573227;
        long long newx1 = (x1 + x2) % 2091573227;
        x1 = sh2.range_hash(l1, r1);
        x2 = B.sh2.range_hash(l2, r2);
        x1 = (x1 * B.sh2.P[len2]) % 2117566807;
        long long newx2 = (x1 + x2) % 2117566807;
        return (newx1 << 32) ^ newx2;
    }
    inline long long range_hash(int l, int r) {
        return ((long long)sh1.range_hash(l, r) << 32)
    ^ sh2.range_hash(l, r);
    }
    inline long long reverse_hash(int l, int r) {
        return ((long long)sh1.reverse_hash(l, r) <<
    32) ^ sh2.reverse_hash(l, r);
    }
};
```

7.3 KMP [23 lines]

```
char P[maxn],T[maxn];
int b[maxn],n,m;
void kmpPreprocess(){
  int i=0,j=-1;
  b[0]=-1;
  while(i<m){
    while(j>=0 and P[i]!=P[j])
      j=b[j];
      i++;j++;
      b[i]=j;
  }
}
void kmpSearch(){
  int i=0,j=0;
  while(i<n){
    while(j>=0 and T[i]!=P[j])
      j=b[j];
      i++;j++;
    if(j==m){
      //pattern found at index i-j
    }
  }
}
```

7.4 Palindromic Tree [30 lines]

```
struct PalindromicTree{
  int n,idx,t;
  vector<vector<int>> tree;
  vector<int> len,link;
  string s; // 1-indexed
  PalindromicTree(string str){
    s="$"+str;
    n=s.size();
    len.assign(n+5,0);
    link.assign(n+5,0);
    tree.assign(n+5,vector<int>(26,0));
  }
  void extend(int p){
    while(s[p-len[t]-1]!=s[p]) t=link[t];
    int x=link[t],c=s[p]-'a';
    while(s[p-len[x]-1]!=s[p]) x=link[x];
    if(!tree[t][c]){
      tree[t][c]=++idx;
      len[idx]=len[t]+2;
      link[idx]=len[idx]==1?2:tree[x][c];
    }
    t=tree[t][c];
  }
  void build(){
    len[1]=-1,link[1]=1;
    len[2]=0,link[2]=1;
    idx=t=2;
    for(int i=1;i<n;i++) extend(i);
  }
};
```

7.5 Suffix Array [78 lines]

```
struct SuffixArray {
vector<int> p, c, rank, lcp;
vector<vector<int>> st;
SuffixArray(string const& s) {
  build_suffix(s + char(1));
  p.erase(p.begin());
  build_rank(p.size());
  build_lcp(s);
  build_sparse_table(lcp.size());
}
void build_suffix(string const& s) {
  int n = s.size();
  const int MX_ASCII = 256;
  vector<int> cnt(max(MX_ASCII, n), 0);
  p.resize(n); c.resize(n);
  for (int i = 0; i < n; i++) cnt[s[i]]++;
  for (int i=1; i<MX_ASCII; i++) cnt[i]+=cnt[i-1];
  for (int i = 0; i < n; i++) p[--cnt[s[i]]] = i;
  c[p[0]] = 0;
  int classes = 1;
  for (int i = 1; i < n; i++) {
    if (s[p[i]] != s[p[i-1]]) classes++;
    c[p[i]] = classes - 1;
  }
  vector<int> pn(n), cn(n);
  for (int h = 0; (1 << h) < n; ++h) {
    for (int i = 0; i < n; i++) {
      pn[i] = p[i] - (1 << h);
      if (pn[i] < 0) pn[i] += n;
    }
    fill(cnt.begin(), cnt.begin() + classes, 0);
    for (int i = 0; i < n; i++) cnt[c[pn[i]]]++;
    for (int i=1; i<classes; i++) cnt[i]+=cnt[i-1];
    for (int i=n-1;i>=0;i--) p[--cnt[c[pn[i]]]]=pn[i];
    cn[p[0]] = 0; classes = 1;
    for (int i = 1; i < n; i++) {
      pair<int, int> cur = {c[p[i]], c[(p[i] + (1 <<
    h)) % n]};
      pair<int, int> prev = {c[p[i-1]], c[(p[i-1] + (1
    << h)) % n]};
      if (cur != prev) ++classes;
      cn[p[i]] = classes - 1;
    }
    c.swap(cn);
  }
}
void build_rank(int n) {
  rank.resize(n, 0);
  for (int i = 0; i < n; i++) rank[p[i]] = i;
}
void build_lcp(string const& s) {
  int n = s.size(), k = 0;
  lcp.resize(n - 1, 0);
  for (int i = 0; i < n; i++) {
    if (rank[i] == n - 1) {
      k = 0;
      continue;
    }
    int j = p[rank[i] + 1];
    while (i + k < n && j + k < n && s[i+k] == s[j+k])
      k++;
    lcp[rank[i]] = k;
    if (k) k--;
  }
}
void build_sparse_table(int n) {
  int lim = __lg(n);
  st.resize(lim + 1, vector<int>(n)); st[0] = lcp;
  for (int k = 1; k <= lim; k++)
    for (int i = 0; i + (1 << k) <= n; i++)
      st[k][i] = min(st[k - 1][i], st[k - 1][i + (1 <<
  (k - 1))]);
}
int get_lcp(int i) { return lcp[i]; }
int get_lcp(int i, int j) {
  if (j < i) swap(i, j);
  j--; /*for lcp from i to j we don't need last lcp*/
  int K = __lg(j - i + 1);
  return min(st[K][i], st[K][j - (1 << K) + 1]);
}
};
```

7.6 Trie [28 lines]

```
const int maxn=100005;
struct Trie{
  int next[27][maxn];
  int endmark[maxn],sz;
  bool created[maxn];
  void insertTrie(string& s){
    int v=0;
    for(int i=0;i<(int)s.size();i++){
      int c=s[i]-'a';
      if(!created[next[c][v]]){
        next[c][v]=++sz;
        created[sz]=true;
      }
      v=next[c][v];
    }
    endmark[v]++;
  }
  bool searchTrie(string& s){
    int v=0;
    for(int i=0;i<(int)s.size();i++){
      int c=s[i]-'a';
      if(!created[next[c][v]])
        return false;
      v=next[c][v];
    }
    return(endmark[v]>0);
  }
};
```

7.7 Z-Algorithm [19 lines]

```
void compute_z_function(const char*S,int N){
  int L=0,R=0;
  for(int i=1;i<N;++i){
    if(i>R){
      L=R=i;
      while(R<N && S[R-L]==S[R])++R;
      Z[i]=R-L,--R;
    }
    else{
      int k=i-L;
      if(Z[k]<R-i+1)Z[i]=Z[k];
      else{
        L=i;
        while(R<N && S[R-k]==S[R])++R;
        Z[i]=R-L,--R;
      }
    }
  }
}
```