Contents

1 Basic 1
1.1 Map 1
2 DataStructure 1
2.1 DisjointSet 1
2.2 SegmentTree 1
3 Graph 1
3.1 Dijkstra 1
4 Math 2
4.1 PrimeTable 2

1 Basic

1.1 Map

```cpp
#include <map>
int main(){

    //declaration container and iterator
    map<string, string> mapStudent;
    map<string, string>::iterator iter;
    map<string, string>::reverse_iterator iter_r;

    //insert element
    mapStudent.insert(pair<string, string>("r000",
        "student_zero"));

    mapStudent["r123"] = "student_first";
    mapStudent["r456"] = "student_second";

    //traversal
    for(iter = mapStudent.begin(); iter !=
        mapStudent.end(); iter++)
                cout<<iter->first<<"
                    "<<iter->second<<endl;
    for(iter_r = mapStudent.rbegin(); iter_r !=
        mapStudent.rend(); iter_r++)
                cout<<iter_r->first<<"
                    "<<iter_r->second<<endl;

    //find and erase the element
    iter = mapStudent.find("r123");
    mapStudent.erase(iter);

    iter = mapStudent.find("r123");

    if(iter != mapStudent.end())
       cout<<"Find, the value is
           "<<iter->second<<endl;
    else
       cout<<"Do not Find"<<endl;

    return 0;
}
```

2 DataStructure

2.1 DisjointSet

```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])); //
      update every child to the other father
}

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

  if(x == y)
    return;

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

2.2 SegmentTree

```cpp
#define SIZE 100000

int st[SIZE];
int st_val[SIZE];

void st_build(int *st, int *st_val, int now, int ls,
    int rs)
{
  if(ls == rs)
    st[now] = st_val[ls];
  else
  {
    st_build(st, st_val, now*2, ls, (ls+rs)/2);
    st_build(st, st_val, now*2+1, (ls+rs)/2+1, rs);
    st[now] = max(st[now*2], st[now*2+1]);
  }
}

// ls and rs are query range, begin and end is whole
    st[] range
int query(int now, int ls, int rs, int begin, int end)
{
  int mid = (begin+end)/2;
  int ret = 0;

  if(ls <= begin && rs >= end)
    return st[now];

  // it is find max now (modify here)
  if(ls <= mid)
    ret = max(ret, query(now*2, ls, rs, begin, mid));

  if(rs > mid)
    ret = max(ret, query(now*2+1, ls, rs, mid+1,
        end));

  return ret;
}
```

3 Graph

3.1 Dijkstra

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

#define MP make_pair
#define PII pair<int, int>
#define maxn 50000+5

int dis[maxn]; // 預設都是 INF
vector<PII > e[maxn]; // (連到的點， 邊的距離)

void dijk(int cur) // dijk(起點)
{
  int d;
  priority_queue<PII,vector<PII>,greater<PII>> q; //
      放 (距離, 點編號)，每次會拿距離最小的點出來
  q.push( MP(0, cur) );

  while (!q.empty())
  {
    tie(d, cur) = q.top();
    q.pop();
    if (dis[cur] != 1e9)
        continue; // 如果之前就拜訪過，無視

    dis[cur] = d;

    for (auto i: e[cur])
        if (dis[i.first] == 1e9)
        {
            q.push( MP(d+i.second, i.first) );
        }
    }
}

void init(void)
{
    fill(dis, dis+maxn, 1e9);

    for(int i = 0; i < maxn; i++)
    {
        e[i].clear();
    }
}
```

4 Math

4.1 PrimeTable

```cpp
void primeTable()
{
    is_notp.reset();
    is_notp[0] = is_notp[1] = 1;
    for (int i = 2; i < N; i++)
    {
        if (!is_notp[i])
        {
            p.push_back(i);
        }
        for (int j = 0; j < (int)p.size() && i * p[j]
            < N; j++)
        {
            is_notp[i * p[j]] = 1;
            if (i % p[j] == 0)
            {
                break;
            }
        }
    }
}
```