Contents

1 Basic

1.1 BIT

```cpp
#define lowbit(k) (k & -k)

int n;
vector<int> B1, B2;

void add(vector<int> &tr, int id, int val) {
  for (; id <= n; id += lowbit(id)) {
    tr[id] += val;
  }
}
void range_add(int l, int r, int val) {
  add(B1, l, val);
  add(B1, r + 1, -val);
  add(B2, l, val * (l - 1));
  add(B2, r + 1, -val * r);
}
int sum(vector<int> &tr, int id) {
  int ret = 0;
  for (; id >= 1; id -= lowbit(id)) {
    ret += tr[id];
  }
  return ret;
}
int prefix_sum(int id) {
  return sum(B1, id) * id - sum(B2, id);
}
int range_sum(int l, int r) {
  return prefix_sum(r) - prefix_sum(l - 1);
}
```

1.2 Black Magic

```cpp
#include <bits/extc++.h>
// #include <ext/pb_ds/assoc_container.hpp>
// #include <ext/pb_ds/tree_policy.hpp>
// #include <ext/pb_ds/priority_queue.hpp>
using namespace std;
using namespace __gnu_pbds;
using set_t =
  tree<int, null_type, less<int>, rb_tree_tag,
    tree_order_statistics_node_update>;//紅黑樹(set)
using map_t =
  tree<int, int, less<int>, rb_tree_tag,
    tree_order_statistics_node_update>;//紅黑樹(map)
using heap_t =
  __gnu_pbds::priority_queue<int>;
using ht_t =
  gp_hash_table<int, int>;
int main() {
  //set-----------------------------------
  set_t st;
  st.insert(5); st.insert(6);
  st.insert(3); st.insert(1);

  // the smallest is (0), biggest is (n-1), kth small is (k-1)
  int num = *st.find_by_order(0);
  cout << num << '\n'; // print 1

  num = *st.find_by_order(st.size() - 1);
  cout << num << '\n'; // print 6

  // find the index
  int index = st.order_of_key(6);//在裡面第幾大
  cout << index << '\n'; // print 3

  // check if there exists x
  int x = 5;
  int check = st.erase(x);
  if (check == 0) printf("st not contain 5\n");
  else if (check == 1) printf("st contain 5\n");

  //tree policy like set
  st.insert(5); st.insert(5);
  cout << st.size() << '\n'; // print 4

  //map-----------------------------------
  map_t mp;
  mp[1] = 2;
  cout << mp[1] << '\n';
  auto tmp = *mp.find_by_order(0); // pair
  cout << tmp.first << " " << tmp.second << '\n';

  //heap----------------------------------
  heap_t h1, h2;
  h1.push(1); h1.push(3);
  h2.push(2); h2.push(4);
  h1.join(h2);
  cout << h1.size() << h2.size() << h1.top() << '\n';
  // 支援合併
  // 404

  //hash-table----------------------------
  ht_t ht;
  ht[85] = 5;
  ht[89975] = 234;
  for (auto i : ht) {
    cout << i.first << " " << i.second << '\n';
  }
  //比較強的unorder map
}
```

1.3 DJS

```cpp
const int MAXN = 1000;
int boss[MAXN];
void init(int n) {
  for (int i = 0; i < n; i++) {
    boss[i] = -1;
  }
}
int find(int x) {
  if (boss[x] < 0) {
    return x;
  }
  return boss[x] = find(boss[x]);
}
bool uni(int a, int b) {
  a = find(a);
  b = find(b);
  if (a == b) {
    return false;
  }
  if (boss[a] > boss[b]) {
    swap(a, b);
  }
  boss[a] += boss[b];
  boss[b] = a;
  return true;
}
```

1.4 ST

```
const int INF = 1e9;
const int MAXN = ;

int n;
int a[MAXN], tr[MAXN << 1];

// !!! remember to call this function
void build() {
  for (int i = 0; i < n; i++) {
    tr[i + n] = a[i];
  }
  for (int i = n - 1; i > 0; i--) {
    tr[i] = max(tr[i << 1], tr[i << 1 | 1]);
  }
}
void update(int id, int val) {
  for (tr[id += n] = val; id > 1; id >>= 1) {
    tr[id >> 1] = max(tr[id], tr[id ^ 1]);
  }
}
int query(int l, int r) { // [l, r)
  int ret = -INF;
  for (l += n, r += n; l < r; l >>= 1, r >>= 1) {
    if (l & 1) {
      ret = max(ret, tr[l++]);
    }
    if (r & 1) {
      ret = max(ret, tr[--r]);
    }
  }
  return ret;
}
```

1.5 DFS

```
struct Edge {
    int bi,color;//a連接到的bi,通道顏色
    bool operator < (const Edge &other) const{
        return color < other.color;
    }
};
vector<Edge>G[maxn];

void DFS(int me,int mydad,int distance){
    if(dist[me] < distance) return;
    dist[me] = distance;
    for(int i = 0;i<G[me].size();i++){
        int v = G[me][i].bi;
        DFS(v,me,distance+1);
    }
}
```

1.6 BFS

```
bool visit[maxn];//訪問過的
void BFS(int point){
    queue<int>q;
    q.push(point);
    while(!q.empty()){
        int u = q.front();
        if(visit[u]) continue;//訪問過就下一個
        visit[u] = true;
        for(int i = 0;i<edge[u][i];i++){//連出去的線丟到queue
            q.push(edge[u][i]);
        }
    }
}
```

2 DP

2.1 LCS

```
int LCS(string s1, string s2) {
  int n1 = s1.size(), n2 = s2.size();
  vector<vector<int>> dp(n1 + 1, vector<int>(n2 + 1, 0));
  for (int i = 1; i <= n1; i++) {
    for (int j = 1; j <= n2; j++) {
      if (s1[i - 1] == s2[j - 1]) {
        dp[i][j] = dp[i - 1][j - 1] + 1;
      } else {
        dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
      }
    }
  }
  return dp[n1][n2];
}
```

2.2 LIS

```
int LIS(vector<int> &a) {
  vector<int> s;
  for (int i = 0; i < a.size(); i++) {
    if (s.empty() || s.back() < a[i]) {
      s.push_back(a[i]);
    } else {
      *lower_bound(s.begin(), s.end(), a[i], [](int x, int y) {return x < y;}) = a[i];
    }
  }
  return s.size();
}
```

2.3 迴文

```
bool isPalindrome[100][100];
// Find the palindromes of a string in O(n^2)

int main()
{
  ios_base::sync_with_stdio(0);
  // freopen("in.txt","r",stdin);
  string s;
  cin>>s;
  int len=s.size();
  for(int i=0; i<len; i++)
    isPalindrome[i][i]=true;

  for(int k=1; k<len; k++){
    for(int i=0; i+k<len; i++){
      int j=i+k;
      isPalindrome[i][j]=(s[i]==s[j]) &&
      (isPalindrome[i+1][j-1] || i+1>=j-1);
    }
  }
  return 0;
}
```

3 Graph

3.1 Bellman Ford

```
bool bellman(int src)
{
    // Nodes are indexed from 1
  for (int i = 1; i <= n; i++)
    dist[i] = INF;
  dist[src] = 0;
    for(int i = 2; i <= n; i++)
    {
        for (int j = 0; j < edges.size(); j++)
        {
            int u = edges[j].first;
            int v = edges[j].second;
            ll weight = adj[u][v];
            if (dist[u]!=INF && dist[u] + weight < dist[v])
                dist[v] = dist[u] + weight;
        }
    }
  for (int i = 0; i < edges.size(); i++)
    {
    int u = edges[i].first;
    int v = edges[i].second;
    ll weight = adj[u][v];
        // True if neg-cylce exists
    if (dist[u]!=INF && dist[u] + weight < dist[v])
      return true;
    }
  return false;
}
```

3.2 Dijk

```
const long long int INF = 1e18;
const int MAXN = 1000000;
struct Edge {
  int to;
  long long int cost;
  Edge(int v, long long int c) : to(v), cost(c) {}
  bool operator < (const Edge &other) const {
    return cost > other.cost;
  }
};

int n;
long long int dis[MAXN];
vector<Edge> G[MAXN];

void init() {
  for (int i = 0; i < n; i++) {
    G[i].clear();
    dis[i] = INF;
  }
}
void Dijkstra(int st, int ed = -1) {
  priority_queue<Edge> pq;
  pq.emplace(st, 0);
  dis[st] = 0;
  while (!pq.empty()) {
    auto now = pq.top();
    pq.pop();
    if (now.to == ed) {
      return;
    }
    if (now.cost > dis[now.to]) {
      continue;
    }
    for (auto &e : G[now.to]) {
      if (dis[e.to] > now.cost + e.cost) {
        dis[e.to] = now.cost + e.cost;
        pq.emplace(e.to, dis[e.to]);
      }
    }
  }
}
```

3.3 Edges

```
struct Edge
{
    int from, to, w;
    bool operator<(const Edge& rhs) // optional
    {
        return w < rhs.w;
    }
};
```

3.4 Floyd

```
const LL INF = 1e18;
const int MAXN = ;

int n;
LL G[MAXN][MAXN];

void init() {
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      G[i][j] = INF;
    }
    G[i][i] = 0;
  }
}
void floyd() {
  for (int k = 0; k < n; k++) {
    for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
        if (G[i][k] != INF && G[k][j] != INF) {
          G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
        }
      }
    }
  }
}
```

3.5 KM

```
const int INF = 1e9;
const int MAXN = ;
struct KM { //1-base
  int n, G[MAXN][MAXN];
  int lx[MAXN], ly[MAXN], my[MAXN];
  bool vx[MAXN], vy[MAXN];
  void init(int _n) {
    n = _n;
    for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= n; j++) {
        G[i][j] = 0;
      }
    }
  }
  bool match(int i) {
    vx[i] = true;
    for (int j = 1; j <= n; j++) {
      if (lx[i] + ly[j] == G[i][j] && !vy[j]) {
        vy[j] = true;
        if (!my[j] || match(my[j])) {
          my[j] = i;
          return true;
        }
      }
    }
    return false;
  }
  void update() {
    int delta = INF;
    for (int i = 1; i <= n; i++) {
      if (vx[i]) {
        for (int j = 1; j <= n; j++) {
          if (!vy[j]) {
            delta = min(delta, lx[i] + ly[j] -
                G[i][j]);
          }
        }
      }
    }
    for (int i = 1; i <= n; i++) {
      if (vx[i]) {
        lx[i] -= delta;
      }
      if (vy[i]) {
        ly[i] += delta;
      }
    }
  }
  int run() {
    for (int i = 1; i <= n; i++) {
      lx[i] = ly[i] = my[i] = 0;
      for (int j = 1; j <= n; j++) {
        lx[i] = max(lx[i], G[i][j]);
      }
    }
    for (int i = 1; i <= n; i++) {
      while (true) {
        for (int i = 1; i <= n; i++) {
          vx[i] = vy[i] = 0;
        }
        if (match(i)) {
          break;
        } else {
          update();
        }
      }
    }
    int ans = 0;
    for (int i = 1; i <= n; i++) {
      ans += lx[i] + ly[i];
    }
    return ans;
  }
};
```

4 Math

4.1 GCDhjackh

```
int extgcd(int a, int b, int c, int &x, int &y) {
  if (b == 0) {
    x = c / a;
    y = 0;
    return a;
  }
  int d = extgcd(b, a % b, c, x, y);
  int tmp = x;
  x = y;
  y = tmp - (a / b) * y;
  return d;
}
```

4.2 Prime

```
const int maxn = ;
int arr[maxn];
int prime[maxn];
void init(){
  for (int i = 0; i < maxn; ++i){
    arr[i] = 0;
  }
}
void find(){
  int num = 0;
  for(int i = 2;i<maxn;i++){
    if(arr[i] == 0){
      prime[num] = 0;
      num++;
      for(int j = i*i;j<maxn;j+=i){
        arr[j] = 1;
      }
    }
  }
}
```

5 String

5.1 KMP

```
void failure(string s, int len, int *f)
{
    f[ 0 ] = -1;
    for(int i = 1; i < len; i++)
    {
        int k = f[ i-1 ];

        while(s[i] != s[k+1] && k >= 0)
            k = f[k];

        if(s[i] == s[k+1])f[i] = k+1;
        else f[i] = -1;
    }
}

int compare(string big, string little, int *f)
{
    int Blen = big.length(), Llen = little.length();
    int i = 0, j = 0;

    while(i < Blen && j < Llen)
    {
        if(big[i] == little[j])
        {
            i++;
            j++;
        }
        else if(j == 0)i++;
        else j = f[j-1] + 1;
    }

    if(j == Llen)return 1;
    else return 0;
}
```

幻欣jp
11
L2
完成準備