the Codebook of LetltCode

Toast 1001

September 29, 2024

1 Template

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back
#define endl '\n'

using namespace std;

int main(){
    IOS;
    return 0;
}
```

2 Basic

2.1 Algorithm

Some useful functions in **"algorithm.h"** and **sorting**.

1. algorithm

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back

```

```cpp
using namespace std;

int main(){
    IOS;
    vector<int> arr = {38, 27, 43, 43, 3, 9, 82, 10};
    int a = 1, b = 2;

    // swap(): swap a and b
    cout << "Before swap: a = " << a << ", b = " << b << endl;
    swap(a, b);
    cout << "After swap: a = " << a << ", b = " << b << endl;

    // min_element(): find the minimum element in the range [first, last)
    cout << "Minimum element in the array: " << *min_element(arr.begin(), arr.end()) << endl;

    // max_element(): find the maximum element in the range [first, last)
    cout << "Maximum element in the array: " << *max_element(arr.begin(), arr.end()) << endl;

    // nth_element(): rearrange the elements in the range [first, last)
    // so that the n-th element is the element that would be in that position in a sorted sequence
    // The other elements are left without any specific order,
    // except that none of the elements preceding nth are greater than it,
    // and none of the elements following it are less
    nth_element(arr.begin(), arr.begin() + 3, arr.end());
    cout << "The 4th smallest element in the array: " << arr[3] << endl;

    // unique():
    cout << "Before unique: ";
    for (int num : arr)
        cout << num << " ";
    cout << endl;

    unique(arr.begin(), arr.end());

    cout << "After unique: ";
```

```
    for (int num : arr)
        cout << num << " ";
    cout << endl;

    // reverse(): reverse the order of the elements in the
        range [first, last)
    cout << "Before reverse: ";
    for (int num : arr)
        cout << num << " ";
    cout << endl;

    reverse(arr.begin(), arr.end());

    cout << "After reverse: ";
    for (int num : arr)
        cout << num << " ";
    cout << endl;

    return 0;
}
```

2. merge sort

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back

using namespace std;

void merge(vector<int>& arr, int l, int mid, int r){
    int n1 = mid - l + 1;
    int n2 = r - mid;

    // create two temporary arrays
    vector<int> larr(n1);
    vector<int> rarr(n2);

    // copy the data to the temporary arrays
    for(int i = 0;i < n1;i++){
        larr[i] = arr[l + i];
    }
```

```
21      for(int i = 0;i < n2;i++){
22          rarr[i] = arr[mid + 1 + i];
23      }
24
25      // merge the two temporary arrays to original array
26      int li = 0, ri = 0, i = l;
27      while(li < n1 && ri < n2){
28          if(larr[li] <= rarr[ri]){
29              arr[i] = larr[li];
30              li++;
31          }else{
32              arr[i] = rarr[ri];
33              ri++;
34          }
35          i++;
36      }
37
38      // remaining elements
39      while(li < n1){
40          arr[i] = larr[li];
41          li++;
42          i++;
43      }
44      while(ri < n2){
45          arr[i] = rarr[ri];
46          ri++;
47          i++;
48      }
49      return;
50  }
51
52  void mergeSort(vector<int> &arr, int l, int r){
53      if(l < r){
54          // Divide the array into two halves
55          int mid = (l + r) / 2;
56          mergeSort(arr, l, mid);
57          mergeSort(arr, mid + 1, r);
58
59          // merge the two halves
60          merge(arr, l, mid, r);
61      }
62
63      return;
```

```
}

int main(){
    IOS;
    vector<int> arr = {38, 27, 43, 3, 9, 82, 10};

    cout << "Original Array: ";
    for (int num : arr)
        cout << num << " ";
    cout << endl;

    // Merge Sort
    mergeSort(arr, 0, arr.size() - 1);

    cout << "Sorted Array: ";
    for (int num : arr)
        cout << num << " ";
    cout << endl;

    return 0;
}
```

3. permutation

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back

using namespace std;

int main(){
    IOS;
    vector<int> arr = {1, 2, 3, 4, 5};

    // is_permutation():
    vector<int> b1 = {1, 5, 4, 3, 2};
    vector<int> b2 = {1, 2, 3, 4, 6};
    cout << is_permutation(arr.begin(), arr.end(), b1.begin())
        << endl; // 1
    cout << is_permutation(arr.begin(), arr.end(), b2.begin())
        << endl; // 0
```

```
18
19      // next_permutation():
20      do{
21          for(int num: arr){
22              cout << num << " ";
23          }
24          cout << endl;
25      }while(next_permutation(arr.begin(), arr.end()));
26      cout << "----------------" << endl;
27
28      // prev_permutation():
29      arr = {5, 4, 3, 2, 1};
30      do{
31          for(int num: arr){
32              cout << num << " ";
33          }
34          cout << endl;
35      }while(prev_permutation(arr.begin(), arr.end()));
36      return 0;
37  }
```

4. selection sort

```
1   #include <bits/stdc++.h>
2
3   #define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
4   #define ll long long
5   #define pb push_back
6
7   using namespace std;
8
9   void selectionSort(vector<int>& arr, int n){
10      for(int i = 0;i < n;i++){
11          swap(arr[i], *min_element(arr.begin() + i, arr.end()));
12      }
13      return;
14  }
15
16  int main(){
17      IOS;
18      vector<int> arr = {38, 27, 43, 3, 9, 82, 10};
19
20      cout << "Orr: ";
```

```
    for (int num : arr)
        cout << num << " ";
    cout << endl;

    // selection sort
    selectionSort(arr, arr.size());

    cout << "Sorted array: ";
    for (int num : arr)
        cout << num << " ";
    cout << endl;
    return 0;
}
```

2.2 stringstream

Something about I/O, likes getline() and stringstream.

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back

using namespace std;

int main(){
    IOS;

    vector<string> words = {"Hello", "World", "from", "C++"};
    string s;
    for(auto &wd: words){
        s += wd + " ";
    }

    stringstream ss(s); // turn string s into stringstream ss
    string word;

    while(ss >> word){
        cout << word << endl;
    }

```

```
    return 0;
}
```

2.3 accumulate

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back

using namespace std;

int main(){
    IOS;

    vector<int> arr = {38, 27, 43, 3, 9, 82, 10};

    // accumulate(): accumulate the elements in the range [first,
       last) using init as the initial value
    int sum = accumulate(arr.begin(), arr.begin() + 4, 0);
    cout << "Sum of the first 4 elements: " << sum << endl;
    return 0;
}
```

2.4 binary search

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back

using namespace std;

int main(){
    IOS;
    vector<int> arr = {1, 5, 3, 4, 2, 6, 7, 9, 8, 10};

    // use functions after sorted
```

[width=0.25]frog.jpg

Figure 1: This frog was uploaded via the file-tree menu.

```
    sort(arr.begin(), arr.end());

    // Binary search
    // lower_bound(): >= val
    int pos1 = *lower_bound(arr.begin(), arr.end(), 5);

    // upper_bound(): > val
    int pos2 = *upper_bound(arr.begin(), arr.end(), 5);

    // < val
    int pos3 = *(lower_bound(arr.begin(), arr.end(), 5) - 1);

    // <= val
    int pos4 = *(upper_bound(arr.begin(), arr.end(), 5) - 1);

    return 0;
}
```

3 Dynamic Programming

3.1 Longest Common Subsequence

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back
#define endl '\n'

using namespace std;

int lcsLen(string s1, string s2, vector<vector<int>> &dp){
    int n = s1.size();
    int m = s2.size();
    for(int i = 0;i < n + 1;i++){
        dp[i][0] = 0;
    }
```

```
16
17      for(int j = 0;j < m + 1;j++){
18          dp[0][j] = 0;
19      }
20
21      for(int i = 1;i < n + 1;i++){
22          for(int j = 1;j < m + 1;j++){
23              if(s1[i - 1] == s2[j - 1]){
24                  dp[i][j] = dp[i - 1][j - 1] + 1;
25              }else{
26                  dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
27              }
28          }
29      }
30      return dp[n][m];
31  }
32
33  string lcs(string s1, string s2, vector<vector<int>> &dp){
34      int n = s1.size();
35      int m = s2.size();
36      int lcslen = lcsLen(s1, s2, dp);
37      string lcsstr = "";
38
39      int i = n, j = m;
40      while(i > 0 && j > 0){
41          if(s1[i - 1] == s2[j - 1]){
42              lcsstr += s1[i - 1];
43              i--;
44              j--;
45          }else{
46              if(dp[i - 1][j] > dp[i][j - 1]){
47                  i--;
48              }else{
49                  j--;
50              }
51          }
52      }
53      reverse(lcsstr.begin(), lcsstr.end());
54      return lcsstr;
55  }
56
57  int main(){
58      IOS;
```

```
    string s1, s2;
    cin >> s1 >> s2;

    int n = s1.size();
    int m = s2.size();
    vector<vector<int>> dp(n + 1, vector<int>(m + 1, 0));

    int lcslen = lcsLen(s1, s2, dp);
    string lcsstr = lcs(s1, s2, dp);
    cout << "Length of LCS: " << lcslen << endl;
    cout << "LCS: " << lcsstr << endl;

    return 0;
}
```

3.2 Longest Increasing Subsequence

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back
#define endl '\n'

using namespace std;

int main(){
    IOS;

    vector<int> a = {10, 22 ,9, 33, 21, 50, 41, 60, 80};

    int n = a.size();
    vector<int> dp(n, 1);
    vector<int> v;
    v.pb(a[0]);
    int lislen = 1;

    // Count the length of LIS
    for(int i = 1;i < n;i++){
        if(a[i] > v.back()){
            v.pb(a[i]);
            dp[i] = v.size();
```

```
            lislen = max(lislen, dp[i]);
        }else{
            auto it = lower_bound(v.begin(), v.end(), a[i]);
            *it = a[i]; // v[it] = a[i]
            dp[i] = it - v.begin() + 1;
        }
    }

    cout << "LIS length: " << lislen << endl;

    // Find LIS
    vector<int> lis;
    for(int i = n - 1;i >= 0;i--){
        if(dp[i] == lislen){
            lis.pb(a[i]);
            lislen--;
        }
    }

    reverse(lis.begin(), lis.end());
    cout << "LIS: ";
    for(int i = 0;i < lis.size();i++){
        cout << lis[i] << " ";
    }
    cout << endl;
    return 0;
}
```

4 Graph

4.1 Dijkstra

Only can use in graph without nagative weights.

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back
#define endl '\n'

using namespace std;
```

```cpp

int main(){
    IOS;
    int n; // number of nodes
    int m; // number of edges

    cin >> n >> m;

    vector<vector<int>> path(n, vector<int>(n, INT_MAX)); // path[i][j]: the distance between node i and node j
    vector<int> dis(n, INT_MAX); // dis[i]: the shortest distance between start node and node i
    vector<bool> vis(n, false); // vis[i]: whether node i is visited

    for(int i = 0;i < m;i++){
        int a, b, d;
        cin >> a >> b >> d;
        path[a - 1][b - 1] = d;
        path[b - 1][a - 1] = d; // if the graph is undirected
    }

    // initialization
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq; // min heap
    pq.push({0, 0});

    // dijkstra
    while(!pq.empty()){
        auto [d, u] = pq.top();
        pq.pop();
        if(vis[u]) continue;
        vis[u] = true;
        dis[u] = d;
        for(int v = 0;v < n;v++){
            if(path[u][v] != INT_MAX && !vis[v]){ // relax
                pq.push({d + path[u][v], v});
            }
        }
    }

    // the distance between start node (node 0) and other nodes
    for(int i = 1;i < n;i++){
        cout << "node " << i << ": " << dis[i] << endl;
```

```
    }
    return 0;
}
```

4.2 Bellman-Ford

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back
#define endl '\n'

using namespace std;

int main(){
    IOS;
    int n; // number of nodes
    int m; // number of edges

    cin >> n >> m;

    vector<tuple<int, int, int>> edges; // edges[i]: {a, b, d}, edge from node a to node b with distance d
    vector<ll> dis(n, INT_MAX); // dis[i]: the shortest distance between start node and node i

    for(int i = 0;i < m;i++){
        int a, b, d;
        cin >> a >> b >> d;
        edges.pb({a, b, d});
        edges.pb({b, a, d}); // if the graph is undirected
    }

    // Initialization
    dis[0] = 0;

    // Bellman-Ford algorithm
    for(int i = 0;i < n;i++){
        bool check = false;
        for(auto &e: edges){
            int a, b, d;
```

```
35              tie(a, b, d) = e;
36              if(dis[a - 1] + d < dis[b - 1]){
37                  dis[b - 1] = dis[a - 1] + d;
38                  check = true;
39              }
40          }
41          if(!check)break;
42      }
43
44      // the distance between start node (node 0) and other nodes
45      for(int i = 1;i < n;i++){
46          cout << "node " << i << ": " << dis[i] << endl;
47      }
48      return 0;
49  }
```

4.3 SPFA

```
1   #include <bits/stdc++.h>
2
3   #define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
4   #define ll long long
5   #define pb push_back
6   #define endl '\n'
7
8   using namespace std;
9
10  int main(){
11      IOS;
12      int n; // number of nodes
13      int m; // number of edges
14
15      cin >> n >> m;
16
17      vector<vector<int>> path(n, vector<int>(n, INT_MAX)); //
            path[i][j]: the distance between node i and node j
18      vector<int> dis(n, INT_MAX); // dis[i]: the shortest distance
            between start node and node i
19
20      for(int i = 0;i < m;i++){
21          int a, b, d;
22          cin >> a >> b >> d;
```

```
23          path[a - 1][b - 1] = d;
24      }
25
26      // initialization
27      queue<int> q;
28      q.push(0);
29      dis[0] = 0;
30
31      // spfa
32      while(!q.empty()){
33          int now = q.front();
34          q.pop();
35
36          for(int i = 0;i < n;i++){
37              if(dis[now] + path[now][i] < dis[i] && path[now][i] !=
                    INT_MAX){
38                  dis[i] = dis[now] + path[now][i];
39                  q.push(i);
40              }
41          }
42      }
43
44      // the distance between start node (node 0) and other nodes
45      for(int i = 1;i < n;i++){
46          cout << "node " << i << ": " << dis[i] << endl;
47      }
48      return 0;
49  }
```

5 Math

5.1 Fast Power

1. Fast pow

```
1   #include <bits/stdc++.h>
2
3   #define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
4   #define ll long long
5   #define pb push_back
6   #define endl '\n'
7
```

```cpp
using namespace std;

ll fastPow(int x, int n){
    if(n == 0) return 1;
    ll ans = fastPow(x, n/2);
    if(n % 2 == 0) return ans * ans;
    return ans * ans * x;
}

int main(){
    IOS;
    cout << fastPow(2, 10);
}
```

2. Fast pow of 2×2 Matrix

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back
#define endl '\n'

using namespace std;

typedef struct _matrix{
    ll c00, c10, c01, c11;
} matrix;

matrix multiply(matrix a, matrix b){
    matrix ans;
    ans.c00 = (a.c00 * b.c00 + a.c01 * b.c10);
    ans.c10 = (a.c10 * b.c00 + a.c11 * b.c10);
    ans.c01 = (a.c00 * b.c01 + a.c01 * b.c11);
    ans.c11 = (a.c10 * b.c01 + a.c11 * b.c11);
    return ans;
}

matrix matrixFastPow(matrix x, int n){
    matrix ans = {1, 0, 0, 1};
    if(n == 0) return ans;
    ans = matrixFastPow(x, n/2);
    if(n % 2 == 0) return multiply(ans, ans);
```

```
    return multiply(multiply(ans, ans), x);
}

int main(){
    IOS;
    matrix x = {1, 1, 1, 0};
    matrix ans = matrixFastPow(x, 5);
    cout << ans.c00 << " " << ans.c01 << endl;
    cout << ans.c10 << " " << ans.c11 << endl;
    return 0;
}
```

3. Fast Fibonacci

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back
#define endl '\n'

using namespace std;

typedef struct _matrix{
    ll c00, c10, c01, c11;
} matrix;

matrix multiply(matrix a, matrix b){
    matrix ans;
    ans.c00 = (a.c00 * b.c00 + a.c01 * b.c10);
    ans.c10 = (a.c10 * b.c00 + a.c11 * b.c10);
    ans.c01 = (a.c00 * b.c01 + a.c01 * b.c11);
    ans.c11 = (a.c10 * b.c01 + a.c11 * b.c11);
    return ans;
}

matrix matrixFastPow(matrix x, int n){
    matrix ans = {1, 0, 0, 1};
    if(n == 0) return ans;
    ans = matrixFastPow(x, n/2);
    if(n % 2 == 0) return multiply(ans, ans);
    return multiply(multiply(ans, ans), x);
}
```

```
30
31 ll fib(int n){
32     if(n == 0)return 0;
33     matrix x = {1, 1, 1, 0};
34     return matrixFastPow(x, n - 1).c00;
35 }
36 int main(){
37     IOS;
38
39     int n;
40     cin >> n;
41     cout << fib(n) << endl;
42     return 0;
43 }
```

5.2 Mod

1. GCD and LCM

```
1  #include <bits/stdc++.h>
2
3  #define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
4  #define ll long long
5  #define pb push_back
6  #define endl '\n'
7
8  using namespace std;
9
10 int gcd(int a, int b){
11     return b == 0 ? a : gcd(b, a % b);
12 }
13
14 int lcm(int a, int b){
15     return a / gcd(a, b) * b;
16 }
17
18 int main(){
19     IOS;
20     int a, b;
21     cin >> a >> b;
22     cout << "GCD: " << gcd(a, b) << endl;
23     cout << "LCM: " << lcm(a, b) << endl;
```

```
    return 0;
}
```

2. eGCD

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back
#define endl '\n'

using namespace std;

int egcd(int a, int b, int &x, int &y){
    if(b == 0){
        x = 1;
        y = 0;
        return a;
    }
    int x1, y1;
    int gcd = egcd(b, a % b, x1, y1);

    // Update x and y
    x = y1;
    y = x1 - (a / b) * y1;

    return gcd;
}

int main(){
    IOS;
    int a, b, x, y;
    cin >> a >> b;
    int gcd = egcd(a, b, x, y);
    cout << "GCD: " << gcd << endl;
    cout << "x: " << x << ", y: " << y << endl;
    cout << "Equation: " << a << "*" << x << " + " << b << "*"
        << y << " = " << gcd << endl;
    return 0;
}
```

5.3 Prime

1. Generating primes

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back
#define endl '\n'

using namespace std;

int main(){
    IOS;
    vector<bool> isPrime(1000001, true);
    isPrime[0] = false;
    isPrime[1] = false;

    for(int i = 2;i < 1000001;i++){
        if(isPrime[i]){
            for(int j = i * 2;j < 1000001;j += i){
                isPrime[j] = false;
            }
        }
    }
    return 0;
}
```

2. Single Prime Judge

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back
#define endl '\n'

using namespace std;

bool prime(int n)
{
    if(n<2) return false;
    if(n<=3) return true;
```

```
    if(!(n%2) || !(n%3)) return false;
    for(int i=5;i*i<=n;i+=6)
        if(!(n%i) || !(n%(i+2))) return false;

    return true;
}

int main(){
    IOS;
    int n;
    cin >> n;
    cout << prime(n) << endl;
    return 0;
}
```

5.4 Binomial Coefficient

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back
#define endl '\n'

using namespace std;

int main(){
    IOS;
    return 0;
}
```

6 Tree

6.1 Segment Tree

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
```

```cpp
#define pb push_back
#define endl '\n'

using namespace std;

static int segment_tree[10000] = {0};

void build(int l, int r, int pos, int arr[]){
    if(l == r){
        segment_tree[pos] = arr[l];
        return;
    }
    int mid = (l + r) / 2;
    build(l, mid, 2 * pos + 1, arr);
    build(mid + 1, r, 2 * pos + 2, arr);
    segment_tree[pos] = segment_tree[2 * pos + 1] + segment_tree[2
        * pos + 2];
}

void modify(int l, int r, int pos, int idx, int val){
    if(l == r){
        segment_tree[pos] = val;
        return;
    }
    int mid = (l + r) / 2;
    if(idx <= mid){
        modify(l, mid, 2 * pos + 1, idx, val);
    }else{
        modify(mid + 1, r, 2 * pos + 2, idx, val);
    }
    segment_tree[pos] = segment_tree[2 * pos + 1] + segment_tree[2
        * pos + 2];
}
```

6.2 Tries

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

#define IOS ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ll long long
#define pb push_back
#define endl '\n'
```

```
7
8  using namespace std;
9
10 class Trie{
11 public:
12     Trie* child[26];
13     int visited = 0;
14     bool isEnd = false;
15 }
16
17 int main(){
18     IOS;
19     string s = "hello";
20     Trie* root = new Trie();
21
22     for(int i = 0; i < s.size(); i++){
23         if(root->child[s[i] - 'a'] == NULL){
24             root->child[s[i] - 'a'] = new Trie();
25         }
26         root = root->child[s[i] - 'a'];
27         root->visited++;
28     }
29     root->isEnd = true;
30
31     return 0;
32 }
```