

Contents

1	Data Structure	
1.1	BIT	
1.2	Segment tree	
1.3	Trie	
2	Divide and Conquer	
2.1	count inversions	
3	DP	
3.1	Josephus	
3.2	LCS	
3.3	LIS	
4	Graph	
4.1	SPFA	
4.2	Dijkstra	
4.3	Floyd Warshall	
4.4	Disjoint set Kruskal	
4.5	KM	
4.6	Dinic	
4.7	Bipartite	
4.8	Hungarian algorithm	
5	Other	
5.1	Bubble Sort Expect Value	
5.2	ORXOR	
5.3	Race to 1	
5.4	X drawing	
5.5	Big Mod	
5.6	Crested Ibis vs Monster	
5.7	dpd Knapsack 1	
5.8	Fraction Floor Sum	
5.9	Homer Simpson	
5.10	Let Me Count The Ways	
5.11	Luggage	
5.12	Number of Pairs	
5.13	SuperSale	
5.14	Walking on the Safe Side	
6	Function	
6.1	strstr	
6.2	substr	
6.3	map set	
6.4	vector	
6.5	setprecision	
6.6	GCD LCM	
6.7	reverse	
6.8	CHAR	
6.9	sort	
6.10	struct	
6.11	deque	
6.12	python template	

1 Data Structure

1.1 BIT

```

1 #define lowbit(k) (k & -k)
2 void add(vector<int> &tr, int id, int val) {
3     for (; id <= n; id += lowbit(id)) {
4         tr[id] += val;
5     }
6 }
7 int sum(vector<int> &tr, int id) {
8     int ret = 0;
9     for (; id >= 1; id -= lowbit(id)) {
10        ret += tr[id];
11    }
12    return ret;
13 }

```

1.2 Segment tree

```

1 int dfs(int lef, int rig){
2     if(lef + 2 == rig){
3         if(num[lef] > num[rig-1]){
4             return lef;
5         }
6         else{
7             return rig-1;
8         }
9     }
10    int mid = (lef + rig)/2;
11    int p1 = dfs(lef, mid);
12    int p2 = dfs(mid, rig);
13    if(num[p1] > num[p2]){
14        return p1;
15    }
16    else{
17        return p2;
18    }
19 }

```

1.3 Trie

```

5
6 1 const int MAXL = ; // 自己填
7 2 const int MAXC = ;
8 3 struct Trie {
9 4     int nex[MAXL][MAXC];
10 5     int len[MAXL];
11 6     int sz;
12 7     void init() {
13 8         memset(nex, 0, sizeof(nex));
14 9         memset(len, 0, sizeof(len));
15 10        sz = 0;
16 11    }
17 12    void insert(const string &str) {
18 13        int p = 0;
19 14        for (char c : str) {
20 15            int id = c - 'a';
21 16            if (!nex[p][id]) {
22 17                nex[p][id] = ++sz;
23 18            }
24 19            p = nex[p][id];
25 20        }
26 21        len[p] = str.length();
27 22    }
28 23    vector<int> find(const string &str, int i) {
29 24        int p = 0;
30 25        vector<int> ans;
31 26        for (; i < str.length(); i++) {
32 27            int id = str[i] - 'a';
33 28            if (!nex[p][id]) {
34 29                return ans;
35 30            }
36 31            p = nex[p][id];
37 32            if (len[p]) {
38 33                ans.pb(len[p]);
39 34            }
40 35        }
41 36        return ans;
42 37    }
43 38 };

```

2 Divide and Conquer

2.1 count inversions

```

1 int arr[maxn], buf[maxn];
2 int count_inversions(int lef, int rig){ // 逆序数对
3     if(rig - lef <= 1) return 0;
4     int mid = (lef + rig)/2;

```

```

5 |     int ans = count_inversions(lef, mid) +
      count_inversions(mid, rig);
6 |     int i = lef, j = mid, k = lef;
7 |     while(i < mid || j < rig){
8 |         if(i >= mid) buf[k] = arr[j++];
9 |         else if(j >= rig) buf[k] = arr[i++];
10 |        else{
11 |            if(arr[i] <= arr[j]) buf[k] = arr[i++];
12 |            else{
13 |                buf[k] = arr[j++];
14 |                ans += mid - i;
15 |            }
16 |        }
17 |        k++;
18 |    }
19 |    for(int k = lef; k < rig; ++k) arr[k] = buf[k];
20 |    return ans;
21 | }

```

3 DP

3.1 Josephus

```

1 | int josephus (int n, int k) {
2 |     // 有 n 個人圍成一圈，每 k 個一次
3 |     return n > 1 ? (josephus(n-1, k) + k) % n : 0;
4 | }
5 | // 回傳最後一人的編號，0 index

```

3.2 LCS

```

1 | int LCS(string s1, string s2) {
2 |     int n1 = s1.size(), n2 = s2.size();
3 |     int dp[n1+1][n2+1] = {0};
4 |     // dp[i][j] = s1的前i個字元和s2的前j個字元
5 |     for (int i = 1; i <= n1; i++) {
6 |         for (int j = 1; j <= n2; j++) {
7 |             if (s1[i - 1] == s2[j - 1]) {
8 |                 dp[i][j] = dp[i - 1][j - 1] + 1;
9 |             } else {
10 |                 dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
11 |             }
12 |         }
13 |     }
14 |     return dp[n1][n2];
15 | }

```

3.3 LIS

```

1 | int LIS(vector<int> &a) { // Longest Increasing
      Subsequence
2 |     vector<int> s;
3 |     for (int i = 0; i < a.size(); i++) {
4 |         if (s.empty() || s.back() < a[i]) {
5 |             s.push_back(a[i]);
6 |         } else {
7 |             *lower_bound(s.begin(), s.end(), a[i],
8 |                 [](int x, int y) {return x < y;}) = a[i];
9 |         }
10 |    }
11 |    return s.size();
12 | }

```

4 Graph

4.1 SPFA

```

1 | bool SPFA(int s){
2 |     // 記得初始化這些陣列
3 |     int cnt[1000+5], dis[1000+5];
4 |     bool inqueue[1000+5];
5 |     queue<int> q;
6 |
7 |     q.push(s);
8 |     dis[s] = 0;
9 |     inqueue[s] = true;
10 |    cnt[s] = 1;
11 |    while(!q.empty()){
12 |        int now = q.front();
13 |        q.pop();
14 |        inqueue[now] = false;
15 |
16 |        for(auto &e : G[now]){
17 |            if(dis[e.t] > dis[now] + e.w){
18 |                dis[e.t] = dis[now] + e.w;
19 |                if(!inqueue[e.t]){
20 |                    cnt[e.t]++;
21 |                    if(cnt[e.t] > m){
22 |                        return false;
23 |                    }
24 |                    inqueue[e.t] = true;
25 |                    q.push(e.t);
26 |                }
27 |            }
28 |        }
29 |    }
30 |    return true;
31 | }

```

4.2 Dijkstra

```

1 | struct Item{
2 |     int u, dis;
3 |     // 取路徑最短
4 |     bool operator < (const Item &other) const{
5 |         return dis > other.dis;
6 |     }
7 | };
8 | int dis[maxn];
9 | vector<Edge> G[maxn];
10 | void dijkstra(int s){
11 |     for(int i = 0; i <= n; i++){
12 |         dis[i] = inf;
13 |     }
14 |     dis[s] = 0;
15 |     priority_queue<Item> pq;
16 |     pq.push({s, 0});
17 |     while(!pq.empty()){
18 |         // 取路徑最短的點
19 |         Item now = pq.top();
20 |         pq.pop();
21 |         if(now.dis > dis[now.u]){
22 |             continue;
23 |         }
24 |         // 鬆弛更新，把與 now.u 相連的點都跑一遍
25 |         for(Edge e : G[now.u]){
26 |             if(dis[e.v] > now.dis + e.w){
27 |                 dis[e.v] = now.dis + e.w;
28 |                 pq.push({e.v, dis[e.v]});
29 |             }
30 |         }
31 |     }
32 | }

```

4.3 Floyd Warshall

```

1 | void floyd_warshall(){
2 |     for(int i = 0; i < n; i++){
3 |         for(int j = 0; j < n; j++){
4 |             G[i][j] = INF;

```

```

5     }
6     G[i][i] = 0;
7 }
8 for (int k = 0; k < n; k++){ //
    嘗試每一個中繼點
9     for (int i = 0; i < n; i++){ //
        計算每一個i點與每一個j點
10        for (int j = 0; j < n; j++){
11            G[i][j] = min(G[i][j], G[i][k] +
12                G[k][j]);
13        }
14    }
15 }

```

4.4 Disjoint set Kruskal

```

1 struct Edge{
2     int u, v, w;
3     // 用權重排序 由大到小
4     bool operator < (const Edge &other) const{
5         return w > other.w;
6     }
7 }edge[maxn];
8 // disjoint set
9 int find(int x){
10     if(parent[x] < 0){
11         return x;
12     }
13     else{
14         return parent[x] = find(parent[x]);
15     }
16 }
17 void unite(int a, int b){
18     a = find(a);
19     b = find(b);
20
21     if(a != b){
22         if(parent[a] < parent[b]){
23             parent[a] += parent[b];
24             parent[b] = a;
25         }
26         else{
27             parent[b] += parent[a];
28             parent[a] = b;
29         }
30     }
31 }
32 void kruskal(){
33     memset(parent, -1, sizeof(parent));
34     sort(edge, edge + m);
35     int i, j;
36     for(i = 0, j = 0; i < n - 1 && j < m; i++){
37         // 如果 u 和 v 的祖先相同，則 j++
38         // (祖先相同代表會產生環 所以不要)
39         while(find(edge[j].u) == find(edge[j].v)) j++;
40         // 若部會產生環 則讓兩點之間產生橋
41         // (連接兩顆子生成樹)
42         unite(edge[j].u, edge[j].v);
43         j++;
44     }
45 }

```

4.5 KM

```

1 const int X = 50; // x的點數，等於y的點數
2 const int Y = 50; // y的點數
3 int adj[X][Y]; // 精簡過的adjacency matrix
4 int lx[X], ly[Y]; // vertex labeling
5 int mx[X], my[Y]; //
    x各點的配對對象、y各點的配對對象
6 int q[X], *qf, *qb; // BFS queue

```

```

7 int p[X]; // BFS
    parent, 交錯樹之偶點，指向上一個偶點
8 bool vx[X], vy[Y]; // 記錄是否在交錯樹上
9 int dy[Y], pdy[Y]; // 表格
10
11 void relax(int x){ // relaxation
12     for (int y=0; y<Y; ++y)
13         if (adj[x][y] != 1e9)
14             if (lx[x] + ly[y] - adj[x][y] < dy[y]){
15                 dy[y] = lx[x] + ly[y] - adj[x][y];
16                 pdy[y] = x; //
                    記錄好是從哪個樹葉連出去的
17             }
18 }
19 void reweight(){ // 調整權重、調整表格
20     int d = 1e9;
21     for (int y=0; y<Y; ++y) if (!vy[y]) d = min(d,
22         dy[y]);
23     for (int x=0; x<X; ++x) if (vx[x]) lx[x] -= d;
24     for (int y=0; y<Y; ++y) if (vy[y]) ly[y] += d;
25     for (int y=0; y<Y; ++y) if (!vy[y]) dy[y] -= d;
26 }
27 void augment(int x, int y){ // 擴充路徑
28     for (int ty; x != -1; x = p[x], y = ty){
29         ty = mx[x]; my[y] = x; mx[x] = y;
30     }
31 }
32 bool branch1(){ // 延展交錯樹：使用既有的等邊
33     while (qf < qb)
34         for (int x=*qf++, y=0; y<Y; ++y)
35             if (!vy[y] && lx[x] + ly[y] == adj[x][y]){
36                 vy[y] = true;
37                 if (my[y] == -1){
38                     augment(x, y);
39                     return true;
40                 }
41                 int z = my[y];
42                 *qb++ = z; p[z] = x; vx[z] = true;
43                 relax(z);
44             }
45     return false;
46 }
47 bool branch2(){ // 延展交錯樹：使用新添的等邊
48     for (int y=0; y<Y; ++y){
49         if (!vy[y] && dy[y] == 0){
50             vy[y] = true;
51             if (my[y] == -1){
52                 augment(pdy[y], y);
53                 return true;
54             }
55             int z = my[y];
56             *qb++ = z; p[z] = pdy[y]; vx[z] = true;
57             relax(z);
58         }
59     }
60     return false;
61 }
62 int Hungarian(){
63     // 初始化vertex labeling
64     // memset(lx, 0, sizeof(lx)); // 任意值皆可
65     memset(ly, 0, sizeof(ly));
66     for (int x=0; x<X; ++x)
67         for (int y=0; y<Y; ++y)
68             lx[x] = max(lx[x], adj[x][y]);
69
70     // x側每一個點，分別建立等邊交錯樹。
71     memset(mx, -1, sizeof(mx));
72     memset(my, -1, sizeof(my));
73     for (int x=0; x<X; ++x){
74         memset(vx, false, sizeof(vx));
75         memset(vy, false, sizeof(vy));
76         memset(dy, 0x7f, sizeof(dy));
77         qf = qb = x;
78         *qb++ = x; p[x] = -1; vx[x] = true; relax(x);
79         while (true){
80             if (branch1()) break;
81             if (branch2()) break;
82         }
83     }
84 }

```

```

78         reweight();
79         if (branch2()) break;
80     }
81 }
82 // 計算最大權完美匹配的權重
83 int weight = 0;
84 for (int x=0; x<X; ++x)
85     weight += adj[x][mx[x]];
86 return weight;
87 }

```

4.6 Dinic

```

1 // Maximum Flow
2 const int V = 100, E = 1000;
3 int adj[V]; // adjacency lists, 初始化為-1。
4 struct Element {int b, r, next;} e[E*2];
5 int en = 0;
6 void addedge(int a, int b, int c){
7     e[en] = (Element){b, c, adj[a]}; adj[a] = en++;
8     e[en] = (Element){a, 0, adj[b]}; adj[b] = en++;
9 }
10 int d[V]; // 最短距離
11 bool visit[V]; // BFS/DFS visit record
12 int q[V]; // queue
13 int BFS(int s, int t){ // 計算最短路徑, 求出容許圖
14     memset(d, 0x7f, sizeof(d));
15     memset(visit, false, sizeof(visit));
16     int qn = 0;
17     d[s] = 0;
18     visit[s] = true;
19     q[qn++] = s;
20
21     for (int qf=0; qf<qn; ++qf){
22         int a = q[qf];
23         for (int i = adj[a]; i != -1; i = e[i].next){
24             int b = e[i].b;
25             if (e[i].r > 0 && !visit[b]){
26                 d[b] = d[a] + 1;
27                 visit[b] = true;
28                 q[qn++] = b;
29                 if (b == t) return d[t];
30             }
31         }
32     }
33     return V;
34 }
35 int DFS(int a, int df, int s, int t){ //
36     求出一條最短擴充路徑, 並擴充流量
37     if (a == t) return df;
38     if (visit[a]) return 0;
39     visit[a] = true;
40     for (int i = adj[a]; i != -1; i = e[i].next){
41         int b = e[i].b;
42         if (e[i].r > 0 && d[a] + 1 == d[b]){
43             int f = DFS(b, min(df, e[i].r), s, t);
44             if (f){
45                 e[i].r -= f;
46                 e[i^1].r += f;
47                 return f;
48             }
49         }
50     }
51     return 0;
52 }
53 int dinitz(int s, int t){
54     int flow = 0;
55     while (BFS(s, t) < V)
56         while (true){
57             memset(visit, false, sizeof(visit));
58             int f = DFS(s, 1e9, s, t);
59             if (!f) break;
60             flow += f;
61         }
62     return flow;
63 }

```

4.7 Bipatirate

```

1 const int maxn = 300 + 5;
2 int n, color[maxn];
3 vector<vector<int>> v(maxn);
4 bool dfs(int s){
5     for(auto it : v[s]){
6         if(color[it] == -1){
7             color[it] = 3 - color[s];
8             if(!dfs(it)){
9                 return false;
10            }
11        }
12        if(color[s] == color[it]){
13            return false;
14        }
15    }
16    return true;
17 }
18 void isBipatirate(){
19     bool flag = true;
20     for(int i = 1; i <= n; ++i){
21         if(color[i] == -1){
22             color[i] = 1;
23             flag &= dfs(i);
24         }
25     }
26     if(flag){
27         cout << "YES" << endl;
28     }
29     else{
30         cout << "NO" << endl;
31     }
32 }
33 int main(){
34     while(cin >> n && n){
35         for(int i = 1; i <= n; ++i) v[i].clear();
36         memset(color, -1, sizeof(color));
37         int a, b;
38         while(cin >> a >> b && (a || b)){
39             v[a].emplace_back(b);
40             v[b].emplace_back(a);
41         }
42         isBipatirate();
43     }
44 }

```

4.8 Hungarian algorithm

```

1 const int maxn = 500+5;
2 int t, N, bn, gn, match[maxn];
3 bool visited[maxn];
4 vector<vector<int>> G(maxn);
5 struct People{
6     int h;
7     string music, sport;
8     People(){}
9     People(int h, string music, string sport){
10         this->h = h;
11         this->music = music;
12         this->sport = sport;
13     }
14 }lef[maxn], rig[maxn];
15 bool check(People boy, People girl){
16     if(abs(boy.h - girl.h) <= 40 && boy.music ==
17         girl.music && boy.sport != girl.sport) return
18         true;
19     return false;
20 }
21 bool dfs(int s){
22     for(int i = 0; i < G[s].size(); ++i){

```

```

21     int v = G[s][i];
22     if(visited[v]) continue;
23     visited[v] = true;
24     if(match[v] == -1 || dfs(match[v])){
25         match[v] = s;
26         return true;
27     }
28 }
29 return false;
30 }
31 int Hungarian(){
32     int cnt = 0;
33     memset(match, -1, sizeof(match));
34     for(int i = 0; i < bn; ++i){
35         memset(visited, false, sizeof(visited));
36         if(dfs(i)) cnt++;
37     }
38     return cnt;
39 }
40 int main(){
41     cin >> t;
42     while(t--){
43         cin >> N;
44         bn = 0, gn = 0;
45         for(int i = 0; i <= N; ++i) G[i].clear();
46         int h;
47         string sex, music, sport;
48         for(int i = 0; i < N; ++i){
49             cin >> h >> sex >> music >> sport;
50             if(sex == "M") lef[bn++] = People(h,
51                 music, sport);
52             else rig[gn++] = People(h, music, sport);
53         }
54         for(int i = 0; i < bn; ++i){
55             for(int j = 0; j < gn; ++j)
56                 if(check(lef[i], rig[j]))
57                     G[i].emplace_back(j);
58         }
59         cout << N - Hungarian() << endl;
60     }
61 }

```

```

1  /* 如何切區段，之所以要1<<n是為了可以跑000~111
2  i = 0, binary i = 000
3  0 : 1 5 7
4  i = 1, binary i = 001
5  1 : 1 5 7
6  i = 2, binary i = 010, 看得出來切了一刀
7  2 : 1 | 5 7
8  i = 3, binary i = 011
9  3 : 1 | 5 7
10 i = 4, binary i = 100, 為了要切在index=2, 所以才要1<<j
11 4 : 1 5 | 7
12 i = 5, binary i = 101
13 5 : 1 5 | 7
14 i = 6, binary i = 110
15 6 : 1 | 5 | 7
16 i = 7, binary i = 111
17 7 : 1 | 5 | 7
18 可以觀察出來，前兩位 bit 是 1 時代表的意義是切在哪裡*/
19 int n;
20 int num[20+7];
21 memset(num, 0, sizeof(num));
22 cin >> n;
23 for(int i = 1; i <= n; i++){
24     cin >> num[i];
25 }
26 int mini = 2147483647; // 不知道為甚麼只有 2147483647
27 給過
28 // 1 << n = n * 2
29 for(int i = 0; i < (1 << n); i++){
30     int XOR = 0, OR = 0;
31     for(int j = 1; j <= n; j++){
32         OR |= num[j];
33         if((i & (1 << j))){
34             XOR ^= OR;
35             OR = 0;
36         }
37         XOR ^= OR;
38         mini = min(mini, XOR);
39     }
40     cout << mini << endl;

```

5 Other

5.1 Bubble Sort Expect Value

```

1  /* 期望值算法：
2  擲一枚公平的六面骰子，其每次「點數」的期望值是 3.5
3  E(x) = 1 * 1/6 + 2 * 1/6 + 3 * 1/6 + 4 * 1/6 + 5 *
4  1/6 + 6 * 1/6
5  = (1 + 2 + 3 + 4 + 5 + 6)/6 = 3.5
6  bubble sort 每兩兩之間交換機率是 1/2
7  總共會做 C(n, 2) 次
8  E(x) = C(n, 2) * 1/2 = (n * (n - 1))/2 * 1/2 */
9  int t, ca = 1;
10 cin >> t;
11 while(t--){
12     long long int n;
13     cin >> n;
14     cout << "Case " << ca++ << ": ";
15     // 如果 (n * (n - 1)) 可以被 4 整除
16     代表最後答案會是整數，否則會是分數
17     if((n * (n - 1)) % 4){
18         cout << ( (n * (n - 1)) / 2 ) << "/2" << endl;
19     }
20     else{
21         cout << ( (n * (n - 1)) / 2 ) / 2 << endl;
22     }
23 }

```

5.2 ORXOR

5.3 Race to 1

```

1  const int N = 1000000;
2  bool sieve[N+5];
3  vector<int> pri;
4  double dp[N+5];
5  void Linear_Sieve(){ // 線性篩
6      for (int i = 2; i < N; i++){
7          if (!sieve[i])
8              pri.push_back(i);
9          for (int p: pri){
10             if (i * p >= N){
11                 break;
12             }
13             sieve[i * p] = true;
14             if (i % p == 0){
15                 break;
16             }
17         }
18     }
19 }
20 double dfs(int n){
21     if(dp[n] != -1) return dp[n];
22     dp[n] = 0;
23     if(n == 1) return dp[n];
24     int total = 0, prime = 0;
25     for(int i = 0; i < pri.size() && pri[i] <= n;
26         i++){
27         total++;
28         if(n % pri[i]) continue;
29         prime++;
30         dp[n] += dfs(n/pri[i]);
31     }
32 }

```

```

31 | dp[n] = (dp[n] + total)/prime; // 算期望值
32 | return dp[n];
33 | }
34 | int main(){
35 |     int t;
36 |     int num;
37 |     int ca = 1;
38 |     for(int i = 0; i <= N; i++){
39 |         dp[i] = -1;
40 |     }
41 |     Linear_Sieve();
42 |     cin >> t;
43 |     while(t--){
44 |         cin >> num;
45 |
46 |         cout << "Case " << ca++ << ": " << fixed <<
47 |             setprecision(10) << dfs(num) << endl;
48 |     }

```

5.4 X drawing

```

1 | long long int n, a, b, p, q, r, s;
2 | cin >> n >> a >> b;
3 | cin >> p >> q >> r >> s;
4 | for(long long int i = p; i <= q; i++){
5 |     for(long long int j = r; j <= s; j++){
6 |         if(abs(i - a) == abs(j - b)){
7 |             cout << '#';
8 |         }
9 |         else{
10 |             cout << '.';
11 |         }
12 |     }
13 |     cout << endl;
14 | }

```

5.5 Big Mod

```

1 | '''
2 | Mod
3 | pow(x, y, z) = x^y % z
4 | '''
5 | # python 如何讀取直到 EOF 用 try except
6 | try:
7 |     while True:
8 |         # input().split() 用空格切開讀取一整行
9 |         # map (型態, input().split()) 才能把值全讀成
10 |             int
11 |         B, P, M = map(int, input().split())
12 |         print(pow(B, P, M))
13 | except EOFError:
14 |     exit

```

5.6 Crested Ibis vs Monster

```

1 | /* dp 背包 - 重量/價值/可重複使用
2 | 因為這題可以重複使用同一條魔法
3 | 所以可以這樣dp*/
4 | int h, n;
5 | cin >> h >> n;
6 | for(int i = 1; i <= n; i++){
7 |     cin >> a[i] >> b[i];
8 | }
9 | memset(dp, 0x3f3f3f3f, sizeof(dp));
10 | dp[0][0] = 0;
11 | for(int i = 1; i <= n; i++){
12 |     for(int j = 0; j <= h; j++){
13 |         dp[i][j] = min(dp[i-1][j], dp[i][max(0, j -
14 |             a[i])] + b[i]);

```

```

15 | }
16 | cout << dp[n][h] << endl;

```

5.7 dpd Knapsack 1

```

1 | // dp 背包 - 時間/數量/價值 - 第幾分鐘符合
2 | int N, W;
3 | cin >> N >> W;
4 | int w[100000+5];
5 | int v[100000+5];
6 | for(int i = 0; i < N; i++){
7 |     cin >> w[i] >> v[i];
8 | }
9 | long long int dp[100000+5];
10 | memset(dp, 0, sizeof(dp));
11 | for(int i = 0; i < N; i++){
12 |     for(int j = W; j >= w[i]; j--){
13 |         dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
14 |     }
15 | }
16 | cout << dp[W] << endl;

```

5.8 Fraction Floor Sum

```

1 | /* [N/i] == M
2 | -> M <= N/i < M + 1
3 | -> N/(M+1) < i <= N/M */
4 | long long int N;
5 | cin >> N;
6 | long long int ans = 0;
7 | for(long long int i = 1; i <= N; i++){
8 |     long long int M = N / i;
9 |     long long int n = N / M;
10 |     // 總共會有 n - i 個的 [N/i] 值都是 M
11 |     ans += (n - i + 1) * M;
12 |     // 更新跳過 以免重複計算
13 |     i = n;
14 | }
15 | cout << ans << endl;

```

5.9 Homer Simpson

```

1 | // dp 背包 - 時間/數量 - 漢堡
2 | int m, n, t;
3 | while(cin >> m >> n >> t){
4 |     int dp[10000+5];
5 |     memset(dp, -1, sizeof(dp));
6 |     dp[0] = 0;
7 |     for(int i = m; i <= t; i++){
8 |         if(dp[i - m] != -1){
9 |             dp[i] = max(dp[i], dp[i - m] + 1);
10 |         }
11 |     }
12 |     for(int i = n; i <= t; i++){
13 |         if(dp[i - n] != -1){
14 |             dp[i] = max(dp[i], dp[i - n] + 1);
15 |         }
16 |     }
17 |     if(dp[t] == -1){ // 時間無法剛好吃滿的時候
18 |         for(int i = t; i >= 0; i--){
19 |             if(dp[i] != -1){
20 |                 cout << dp[i] << " " << t - i << endl;
21 |                 break;
22 |             }
23 |         }
24 |     }
25 |     else{
26 |         cout << dp[t] << endl;
27 |     }
28 | }

```

5.10 Let Me Count The Ways

```

1 // dp - 時間/數量 - 硬幣排序
2 long long int n, dp[30000+5];
3 int coin[] = {1, 5, 10, 25, 50};
4 memset(dp, 0, sizeof(dp));
5 dp[0] = 1;
6 for(int i = 0; i < 5; i++){
7     for(int j = coin[i]; j < 30000+5; j++){
8         if(dp[j - coin[i]] != -1){
9             dp[j] += dp[j - coin[i]];
10        }
11    }
12 }
13 while(cin >> n){
14     if(dp[n] == 1){
15         cout << "There is only " << dp[n] << " way to
16             produce " << n << " cents change." <<
17             endl;
18     }
19     else{
20         cout << "There are " << dp[n] << " ways to
21             produce " << n << " cents change." <<
22             endl;
23     }
24 }

```

5.11 Luggage

```

1 // dp 背包 - 重量/是否成立
2 int t;
3 cin >> t;
4 cin.ignore();
5 while(t--){
6     string str;
7     getline(cin, str);
8     vector<int> v;
9     stringstream ss;
10    int num, cnt = 0, sum = 0;;
11    bool dp[4000+5];
12    memset(dp, false, sizeof(dp));
13    ss << str;
14    while(ss >> num){
15        cnt++;
16        sum += num;
17        v.emplace_back(num);
18    }
19    if(sum & 1){
20        cout << "NO" << endl;
21        continue;
22    }
23    dp[0] = true;
24    for(int i = 0; i < v.size(); i++){
25        for(int j = sum; j >= v[i]; j--){
26            if(dp[j - v[i]]){
27                dp[j] = true;
28            }
29        }
30    }
31    cout << (dp[sum/2] ? "YES" : "NO") << endl;
32 }

```

5.12 Number of Pairs

```

1 /* upper_bound ex:
2 10 20 30 30 40 50
3 upper_bound for element 30 is at index 4
4 lower_bound ex:
5 10 20 30 40 50
6 lower_bound for element 30 at index 2 */
7 int t;
8 cin >> t;
9 while(t--){

```

```

10     int n, l, r;
11     vector<int> v;
12     cin >> n >> l >> r;
13     int num;
14     for(int i = 0; i < n; i++){
15         cin >> num;
16         v.emplace_back(num);
17     }
18     sort(v.begin(), v.end());
19     long long int ans = 0;
20     for(int i = 0; i < n; i++){
21         ans += (upper_bound(v.begin() + i + 1,
22                             v.end(), r - v[i]) -
23                 lower_bound(v.begin() + i + 1, v.end(), l
24                             - v[i]));
25     }
26     cout << ans << endl;
27 }

```

5.13 SuperSale

```

1 // dp 背包 - 重量/價值/不可重複使用 - 舉重
2 int t;
3 cin >> t;
4 while(t--){
5     int n;
6     cin >> n;
7     for(int i = 0; i < n; i++){
8         cin >> edge[i].p >> edge[i].w;
9     }
10    int g, total = 0;
11    cin >> g;
12    for(int i = 0; i < g; i++){
13        int pw, dp[30+5];
14        cin >> pw;
15        memset(dp, 0, sizeof(dp));
16        for(int j = 0; j < n; j++){
17            for(int k = pw; k >= edge[j].w; k--){
18                dp[k] = max(dp[k], dp[k - edge[j].w]
19                            + edge[j].p);
20            }
21        }
22        total += dp[pw];
23    }
24    cout << total << endl;
25 }

```

5.14 Walking on the Safe Side

```

1 // dp - 地圖更新
2 int t;
3 bool space = false;
4 cin >> t;
5 while(t--){
6     if(space){
7         cout << endl;
8     }
9     else{
10        space = true;
11    }
12    int r, c;
13    cin >> r >> c;
14    cin.ignore();
15    memset(mp, false, sizeof(mp));
16    memset(dp, 0, sizeof(dp));
17    string str;
18    for(int i = 0; i < r; i++){
19        getline(cin, str);
20        int n, num;
21        stringstream ss(str);
22        ss >> n;
23        while(ss >> num){
24            mp[n][num] = true;
25        }
26    }
27 }

```

```

25     }
26 }
27 dp[1][1] = 1;
28 for(int i = 1; i <= r; i++){
29     for(int j = 1; j <= c; j++){
30         if(mp[i][j]){
31             continue;
32         }
33         if(i > 1){
34             dp[i][j] += dp[i-1][j];
35         }
36         if(j > 1){
37             dp[i][j] += dp[i][j-1];
38         }
39     }
40 }
41 cout << dp[r][c] << endl;
42 }

```

6 Function

6.1 strstr

```

1 #include <stdio.h>
2 #include <string.h>
3
4 int main(){
5     char * c;
6     char str1[1005], str2[1005];
7     scanf("%s %s", str1, str2);
8     c = strstr(str1, str2);
9     if (c != NULL){
10         printf("Yes\n");
11     }
12     else printf("No\n");
13 }
14 // Input : Hello eLL
15 // Output : No

```

6.2 substr

```

1 int main(){
2     string str; //abcdef
3     cin >> str;
4     string tmp;
5     tmp = str.substr(0, 2); //ab
6     str = str.substr(2); //cdef
7     cout << tmp << " " << str;
8     return 0;
9 }

```

6.3 map set

```

1 .begin( ) // Return iterator to beginning
2 .end( ) // Return iterator to end
3 .empty( ) // 檢查是否為空
4 .size( ) // 回傳大小
5 mp.insert(pair<char, int>('a', 100))
6 st.insert(100) // 插入key、value
7 .erase( ) // 刪掉指定key和他的value
8 .clear( ) // 清空整個 map
9 m.find( )
10 cout << "a => " << mymap.find('a')->second << endl;
11 // 找出 map 裡 key
12 // 有沒有在裡面，如果有的話會回傳元素所在的 iterator
13 s.count() // 返回某個值元素在set的個數
14 while( !mymap.empty()){
15     cout << mymap.begin()->first << " => " <<
16     mymap.begin()->second << endl;

```

```

15 mymap.erase(mymap.begin());
16 }
17 for (auto it = mymap.begin(); it != mymap.end(); ++it)
18     cout << it->first << " => " << it->second << endl;

```

6.4 vector

```

1 v.erase(v.begin() + 5) //拿掉第六個數
2 v.erase (v.begin(), v.begin() + 3); //拿掉前三個數

```

6.5 setprecision

```

1 // 將數字的小數部分設定為固定長度
2 cnt = 3.5555;
3 cout << fixed << setprecision(3) << cnt ;
4 // output : 3.555

```

6.6 GCD LCM

```

1 int gcd(int a, int b){
2     return (b == 0 ? a : gcd(b, a % b));
3 }
4 int lcm(int a, int b){
5     return a * b / gcd(a, b);
6 }
7
8 /* 輾轉相除法 - 求兩數是否互質
9 如果兩數互質 最終結果其中一方為0時 另一方必為1
10 若兩數有公因數 最終結果其中一方為0時 另一方必不為1 */
11 while ( ( num1 % = num2 ) != 0 && ( num2 % = num1 ) !=
12         0 );

```

6.7 reverse

```

1 int a[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
2 reverse(a, a+5) // 轉換0~5
3
4 vector<int> v;
5 reverse(v.begin(), v.end());
6
7 string str = "123";
8 reverse(str.begin(), str.end());
9 cout << str << endl; //321

```

6.8 CHAR

```

1 isdigit()
2 isalnum() //判斷字母 // 數字
3 isalpha()
4 islower()
5 isupper()
6 isblank() //判斷是否為空格，或者 tab 健制表符，即
7 // space 和 \t
8 toupper()
9 tolower()

```

6.9 sort

```

1 priority_queue<int, vector<int>, less<int>> //大到小
2 priority_queue<int, vector<int>, greater<int>>
3 //小到大
4 int arr[] = {4, 5, 8, 3, 7, 1, 2, 6, 10, 9};
5 sort(arr, arr+10);

```



```
6|
7| vector<int> v;
8| sort(v.begin(), v.end()); //小到大
9|
10| int cmp(int a, int b){
11|     return a > b;
12| }
13| sort(v.begin(), v.end(), cmp); //大到小
```

6.10 struct

```
1| struct area{
2|     int a, b;
3|     bool operator<(const area rhs) const{
4|         return a > rhs.a || ( a == a && b > rhs.b);
5|     }
6|     bool operator!=(const area rhs) const{
7|         return a != rhs.a || b != rhs.b;
8|     }
9| };
```

6.11 deque

```
1| deque <int> que;
2| que.push_back(10);
3| que.push_front(20);
4| que.front()
5| que.back()
6| que.pop_front()
7| que.pop_back()
8| cout << "Element at position 2 : " << que.at(2) <<
   endl;
```

6.12 python template

```
1| import math
2| import operator
3|
4| try:
5|     while(1):
6|         listx = []
7|         listx.append("...")
8|         list_s = sorted(listx) # 小到大
9|         list_s = sorted(listx, reverse = True) #
           大到小
10|         # max(listx)
11|         # min(listx)
12|         # sum(listx)
13|         # len(listx)
14|         dicty = {}
15|         dicty[key] = "value"
16|         dicty= sorted(dicty.items()) # by key
17|         dicty= sorted(dicty.items(),
           key=operator.itemgetter(1)) # by value
18|         # EOF寫法
19|         # 階層 math.factorial(3) == 6
20|         # 絕對值 math.fabs(x)
21|         # 無條件進位 math.ceil(3.1) == 3
22|         # 無條件捨去 math.floor(2.9) == 2
23|         # C n 取 k math.comb(n, k)
24|         # math.gcd
25|         # math.lcm
26|         # e 次 x 幂 math.exp(x)
27| except EOFError:
28|     pass
```