Contents

1 Data Structure

1.1 BIT

```cpp
#define lowbit(k) (k & -k)
void add(vector<int> &tr, int id, int val) {
  for (; id <= n; id += lowbit(id)) {
    tr[id] += val;
  }
}
int sum(vector<int> &tr, int id) {
  int ret = 0;
  for (; id >= 1; id -= lowbit(id)) {
    ret += tr[id];
  }
  return ret;
}
```

1.2 Segment tree

```cpp
int dfs(int lef, int rig){
    if(lef + 2 == rig){
        if(num[lef] > num[rig-1]){
            return lef;
        }
        else{
            return rig-1;
        }
    }
    int mid = (lef + rig)/2;
    int p1 = dfs(lef, mid);
    int p2 = dfs(mid, rig);
    if(num[p1] > num[p2]){
        return p1;
    }
    else{
        return p2;
    }
}
```

1.3 Trie

```cpp
const int MAXL = ; // 自己填
const int MAXC = ;
struct Trie {
  int nex[MAXL][MAXC];
  int len[MAXL];
  int sz;
  void init() {
    memset(nex, 0, sizeof(nex));
    memset(len, 0, sizeof(len));
    sz = 0;
  }
  void insert(const string &str) {
    int p = 0;
    for (char c : str) {
      int id = c - 'a';
      if (!nex[p][id]) {
        nex[p][id] = ++sz;
      }
      p = nex[p][id];
    }
    len[p] = str.length();
  }
  vector<int> find(const string &str, int i) {
    int p = 0;
    vector<int> ans;
    for (; i < str.length(); i++) {
      int id = str[i] - 'a';
      if (!nex[p][id]) {
        return ans;
      }
      p = nex[p][id];
      if (len[p]) {
        ans.pb(len[p]);
      }
    }
    return ans;
  }
};
```

2 Divide and Conquer

2.1 count inversions

```cpp
int arr[maxn], buf[maxn];
int count_inversions(int lef, int rig){ // 逆序數對
    if(rig - lef <= 1) return 0;
    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 Doubling

```
1  /* 倍增 */
2  int LOG = sqrt(N); // 2^LOG >= N
3  vector<int> arr(N);
4  vector<vector<int>> dp(N, vector<int>(LOG));
5  for(int i = 0; i < N; ++i) cin >> arr[i];
6  int L, Q, a, b;
7  cin >> L >> Q;
8  for(int i = 0; i < N; ++i){
9      dp[i][0] = lower_bound(arr.begin(), arr.end(),
           arr[i] + L) - arr.begin();
10     if(dp[i][0] == N || arr[i] + L < arr[dp[i][0]])
           dp[i][0] -= 1;
11 }
12 for(int i = 1; i < LOG; ++i)
13     for(int j = 0; j < N; ++j)
14         dp[j][i] = dp[dp[j][i - 1]][i - 1];
15 for(int i = 0; i < Q; ++i){
16     cin >> a >> b;
17     a--; // 要減減是因為arr的index從0開始但題目從1開始
18     b--;
19     if(a > b) swap(a, b);
20     int ans = 0;
21     for(int i = LOG - 1; i >= 0; --i){ // 從後往回推
22         if(dp[a][i] < b){
23             ans += (1 << i);
24             a = dp[a][i];
25         }
26     }
27     cout << ans + 1 << endl;
28 }
```

3.2 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
```

3.3 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 }
```

3.4 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 Enumerate

4.1 Halfcut Enumerate

```
1  /* 折半枚舉 */
2  void dfs(set<long long int> &s, int depth, int T,
       long long int sum){
3      if(depth >= T){
4          s.insert(sum);
5          return;
6      }
7      dfs(s, depth + 1, T, sum); // 取或不取的概念
8      dfs(s, depth + 1, T, sum + A[depth]);
9  }
10 int main(){
11     int N, T;
12     set<long long int> s1, s2;
13     cin >> N >> T;
14     for(int i = 0; i < N; ++i) cin >> A[i];
15     dfs(s1, 0, N/2, 0); // 折半枚舉
16     dfs(s2, N/2, N, 0);
17     long long int ans = 0;
18     // 題目:枚舉集合 Sx 的數字 Sxi，找出 Sy
           集合內小於等於 T-Sxi 中最大的數 Syj
19     for(auto &x : s1){
20         auto it = s2.upper_bound(T - x);
21         long long int y = *(--it);
22         if(x + y <= T) ans = max(ans, x + y);
23     }
24     cout << ans << endl;
25 }
```

5 Graph

5.1 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 }
```

5.2 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 }
```

5.3 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] +
                       G[k][j]);
12             }
13         }
14     }
15 }
```

5.4 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++
               (祖先相同代表會產生環  所以不要)
38         while(find(edge[j].u) == find(edge[j].v)) j++;
39         // 若部會產生環  則讓兩點之間產生橋
               (連接兩顆子生成樹)
40         unite(edge[j].u, edge[j].v);
41         j++;
42     }
43 }
```

5.5 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,
            dy[y]);
22      for (int x=0; x<X; ++x) if ( vx[x]) lx[x] -= d;
23      for (int y=0; y<Y; ++y) if ( vy[y]) ly[y] += d;
24      for (int y=0; y<Y; ++y) if (!vy[y]) dy[y] -= d;
25  }
26  void augment(int x, int y){ // 擴充路徑
27      for (int ty; x != -1; x = p[x], y = ty){
28          ty = mx[x]; my[y] = x; mx[x] = y;
29      }
30  }
31  bool branch1(){ // 延展交錯樹：使用既有的等邊
32      while (qf < qb)
33          for (int x=*qf++, y=0; y<Y; ++y)
34              if (!vy[y] && lx[x] + ly[y] == adj[x][y]){
35                  vy[y] = true;
36                  if (my[y] == -1){
37                      augment(x, y);
38                      return true;
39                  }
40                  int z = my[y];
41                  *qb++ = z; p[z] = x; vx[z] = true;
                        relax(z);
42              }
43      return false;
44  }
45  bool branch2(){ // 延展交錯樹：使用新添的等邊
46      for (int y=0; y<Y; ++y)
47          if (!vy[y] && dy[y] == 0){
48              vy[y] = true;
49              if (my[y] == -1){
50                  augment(pdy[y], y);
51                  return true;
52              }
53              int z = my[y];
54              *qb++ = z; p[z] = pdy[y]; vx[z] = true;
                    relax(z);
55          }
56      }
57      return false;
58  }
59  int Hungarian(){
60      // 初始化 vertex labeling
61      // memset(lx, 0, sizeof(lx));  // 任意值皆可
62      memset(ly, 0, sizeof(ly));
63      for (int x=0; x<X; ++x)
64          for (int y=0; y<Y; ++y)
65              lx[x] = max(lx[x], adj[x][y]);
66
67      // X側每一個點，分別建立等邊交錯樹。
68      memset(mx, -1, sizeof(mx));
69      memset(my, -1, sizeof(my));
70      for (int x=0; x<X; ++x){
71          memset(vx, false, sizeof(vx));
72          memset(vy, false, sizeof(vy));
73          memset(dy, 0x7f, sizeof(dy));
74          qf = qb = q;
75          *qb++ = x; p[x] = -1; vx[x] = true; relax(x);
76          while (true){
77              if (branch1()) break;
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  }
```

5.6 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){ //
            求出一條最短擴充路徑，並擴充流量
36      if (a == t) return df;
37      if (visit[a]) return 0;
38      visit[a] = true;
39      for (int i = adj[a]; i != -1; i = e[i].next){
40          int b = e[i].b;
41          if (e[i].r > 0 && d[a] + 1 == d[b]){
42              int f = DFS(b, min(df, e[i].r), s, t);
43              if (f){
44                  e[i].r -= f;
45                  e[i^1].r += f;
46                  return f;
47              }
48          }
49      }
50      return 0;
51  }
52  int dinitz(int s, int t){
53      int flow = 0;
54      while (BFS(s, t) < V)
55          while (true){
56              memset(visit, false, sizeof(visit));
57              int f = DFS(s, 1e9, s, t);
58              if (!f) break;
59              flow += f;
60          }
61      return flow;
62  }
```

5.7 Bipatirate

```cpp
const int maxn = 300 + 5;
int n, color[maxn];
vector<vector<int>> v(maxn);
bool dfs(int s){
    for(auto it : v[s]){
        if(color[it] == -1){
            color[it] = 3 - color[s];
            if(!dfs(it)){
                return false;
            }
        }
        if(color[s] == color[it]){
            return false;
        }
    }
    return true;
}
void isBipatirate(){
    bool flag = true;
    for(int i = 1; i <= n; ++i){
        if(color[i] == -1){
            color[i] = 1;
            flag &= dfs(i);
        }
    }
    if(flag){
        cout << "YES" << endl;
    }
    else{
        cout << "NO" << endl;
    }
}
int main(){
    while(cin >> n && n){
        for(int i = 1; i <= n; ++i) v[i].clear();
        memset(color, -1, sizeof(color));
        int a, b;
        while(cin >> a >> b && (a || b)){
            v[a].emplace_back(b);
            v[b].emplace_back(a);
        }
        isBipatirate();
    }
}
```

5.8 Hungarian algorithm

```cpp
const int maxn = 500+5;
int t, N, bn, gn, match[maxn];
bool visited[maxn];
vector<vector<int>> G(maxn);
struct People{
    int h;
    string music, sport;
    People(){}
    People(int h, string music, string sport){
        this->h = h;
        this->music = music;
        this->sport = sport;
    }
}lef[maxn], rig[maxn];
bool check(People boy, People girl){
    if(abs(boy.h - girl.h) <= 40 && boy.music ==
        girl.music && boy.sport != girl.sport) return
        true;
    return false;
}
bool dfs(int s){
    for(int i = 0; i < G[s].size(); ++i){
        int v = G[s][i];
        if(visited[v]) continue;
        visited[v] = true;
        if(match[v] == -1 || dfs(match[v])){
            match[v] = s;
            return true;
        }
    }
    return false;
}
int Hungarian(){
    int cnt = 0;
    memset(match, -1, sizeof(match));
    for(int i = 0; i < bn; ++i){
        memset(visited, false, sizeof(visited));
        if(dfs(i)) cnt++;
    }
    return cnt;
}
int main(){
    cin >> t;
    while(t--){
        cin >> N;
        bn = 0, gn = 0;
        for(int i = 0; i <= N; ++i) G[i].clear();
        int h;
        string sex, music, sport;
        for(int i = 0; i < N; ++i){
            cin >> h >> sex >> music >> sport;
            if(sex == "M") lef[bn++] = People(h,
                music, sport);
            else rig[gn++] = People(h, music, sport);
        }
        for(int i = 0; i < bn; ++i){
            for(int j = 0; j < gn; ++j)
                if(check(lef[i], rig[j]))
                    G[i].emplace_back(j);
        }
        cout << N - Hungarian() << endl;
    }
}
```

6   Other

6.1 Bubble Sort Expect Value

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

6.2 ORXOR

```cpp
/* 如何切區段，之所以要1<<n是為了可以跑000~111
i = 0，binary i = 000
0 : 1 5 7
i = 1，binary i = 001
1 : 1 5 7
i = 2，binary i = 010，看得出來切了一刀
```

```cpp
 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
        給過
27  // 1 << n = n * 2
28  for(int i = 0; i < (1 << n); i++){
29      int XOR = 0, OR = 0;
30      for(int j = 1; j <= n; j++){
31          OR |= num[j];
32          if((i & (1 << j))){
33              XOR ^= OR;
34              OR = 0;
35          }
36      }
37      XOR ^= OR;
38      mini = min(mini, XOR);
39  }
40  cout << mini << endl;
```

6.3 Race to 1

```cpp
 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;
            i++){
26          total++;
27          if(n % pri[i]) continue;
28          prime++;
29          dp[n] += dfs(n/pri[i]);
30      }
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;
```

```cpp
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 <<
                setprecision(10) << dfs(num) << endl;
47      }
48  }
```

6.4 X drawing

```cpp
 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  }
```

6.5 Big Mod

```python
 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
                int
10          B, P, M = map(int, input().split())
11          print(pow(B, P, M))
12  except EOFError:
13      exit
```

6.6 Crested Ibis vs Monster

```cpp
 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 -
                a[i])] + b[i]);
14      }
15  }
16  cout << dp[n][h] << endl;
```

6.7 dpd Knapsack 1

```cpp
// dp 背包 - 時間/數量/價值 - 第幾分鐘符合
int N, W;
cin >> N >> W;
int w[100000+5];
int v[100000+5];
for(int i = 0; i < N; i++){
    cin >> w[i] >> v[i];
}
long long int dp[100000+5];
memset(dp, 0, sizeof(dp));
for(int i = 0; i < N; i++){
    for(int j = W; j >= w[i]; j--){
        dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
    }
}
cout << dp[W] << endl;
```

6.8 Fraction Floor Sum

```cpp
/* [N/i] == M
-> M <= N/i < M + 1
-> N/(M+1) < i <= N/M */
long long int N;
cin >> N;
long long int ans = 0;
for(long long int i = 1; i <= N; i++){
    long long int M = N / i;
    long long int n = N / M;
    // 總共會有 n - i 個的 [N/i] 值都是 M
    ans += (n - i + 1) * M;
    // 更新跳過 以免重複計算
    i = n;
}
cout << ans << endl;
```

6.9 Homer Simpson

```cpp
// dp 背包 - 時間/數量 - 漢堡
int m, n, t;
while(cin >> m >> n >> t){
    int dp[10000+5];
    memset(dp, -1, sizeof(dp));
    dp[0] = 0;
    for(int i = m; i <= t; i++){
        if(dp[i - m] != -1){
            dp[i] = max(dp[i], dp[i - m] + 1);
        }
    }
    for(int i = n; i <= t; i++){
        if(dp[i - n] != -1){
            dp[i] = max(dp[i], dp[i - n] + 1);
        }
    }
    if(dp[t] == -1){ // 時間無法剛好吃滿的時候
        for(int i = t; i >= 0; i--){
            if(dp[i] != -1){
                cout << dp[i] << " " << t - i << endl;
                break;
            }
        }
    }
    else{
        cout << dp[t] << endl;
    }
}
```

6.10 Let Me Count The Ways

```cpp
// dp - 時間/數量 - 硬幣排序
long long int n, dp[30000+5];
int coin[] = {1, 5, 10, 25, 50};
memset(dp, 0, sizeof(dp));
dp[0] = 1;
for(int i = 0; i < 5; i++){
    for(int j = coin[i]; j < 30000+5; j++){
        if(dp[j - coin[i]] != -1){
            dp[j] += dp[j - coin[i]];
        }
    }
}
while(cin >> n){
    if(dp[n] == 1){
        cout << "There is only " << dp[n] << " way to
            produce " << n << " cents change." <<
            endl;
    }
    else{
        cout << "There are " << dp[n] << " ways to
            produce " << n << " cents change." <<
            endl;
    }
}
```

6.11 Luggage

```cpp
// dp 背包 - 重量/是否成立
int t;
cin >> t;
cin.ignore();
while(t--){
    string str;
    getline(cin , str);
    vector<int> v;
    stringstream ss;
    int num, cnt = 0, sum = 0;;
    bool dp[4000+5];
    memset(dp, false, sizeof(dp));
    ss << str;
    while(ss >> num){
        cnt++;
        sum += num;
        v.emplace_back(num);
    }
    if(sum & 1){
        cout << "NO" << endl;
        continue;
    }
    dp[0] = true;
    for(int i = 0; i < v.size(); i++){
        for(int j = sum; j >= v[i]; j--){
            if(dp[j - v[i]]){
                dp[j] = true;
            }
        }
    }
    cout << (dp[sum/2] ? "YES" : "NO") << endl;
}
```

6.12 Number of Pairs

```cpp
/* uper_bound ex:
10 20 30 30 40 50
upper_bound for element 30 is at index 4
lower_bound ex:
10 20 30 40 50
lower_bound for element 30 at index 2 */
int t;
cin >> t;
while(t--){
    int n, l, r;
    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,
                    v.end(), r - v[i]) -
                    lower_bound(v.begin() + i + 1, v.end(), l
                    - v[i]));
22      }
23      cout << ans << endl;
24 }
```

```
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 }
```

6.13 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
                            + edge[j].p);
19             }
20         }
21         total += dp[pw];
22     }
23     cout << total << endl;
24 }
```

6.14 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     }
```

7 Function

7.1 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
```

7.2 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  }
```

7.3 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
          有沒有在裡面，如果有的話會回傳元素所在的 iterator，否則傳
12 s.count() // 返回某個值元素在 set的個數
13 while( !mymap.empty()){
14     cout << mymap.begin()->first << " => " <<
           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;
```

7.4 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
```

7.5 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
```

7.6 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 ) != 
       0 );
```

7.7 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
```

7.8 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
       space 和 \t
7 toupper()
8 tolower()
```

7.9 sort

```
1 priority_queue<int, vector<int>, less<int>> //大到小
2 priority_queue<int, vector<int>, greater<int>>
       //小到大
3
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
```

7.10 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 };
```

7.11 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;
```

7.12 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 冪 math.exp(x)
27 except EOFError:
28     pass
```