Dynamic Programming

- Matrix Multiplication
  - UVa 10891 - Game of Sum
  - UVa 10003 - Cutting Sticks
- Optimal Search Tree
  - UVa 10304 - Optimal Binary Search Tree
- Integer partition
  - UVa 907 - Winterim Backpacking Trip
- Longest Common Subsequence
  - UVa 10066 - The Twin Towers
  - UVa 10192 - Vacation
  - UVa 10723 - Cyborg Genes
  - UVa 11151 - Longest Palindrome
  - UVa 12147 - DNA Sequences
- Longest Common Substring
  - UVa 1223 - Editor
- Edit Distance
  - UVa 10739 - String to Palindrome
  - UVa 1207 - AGTC
- Maximum Sum Contiguous Subsequence
  - UVa 10684 - The Jackpot
  - UVa 11059 - Maximum Product
- Maximum Sum Sub-rectangle
  - UVa 10827 - Maximum sum on a torus
- Longest Increasing Subsequence
  - UVa 473 - Raucous Rockers
  - UVa 11003 - Boxes
  - UVa 10635 - Prince and Princess
  - UVa 10154 - Weights and Measures
  - UVa 10051 - Tower of Cubes
- Memoized DFS
  - UVa 10259 - Hippity Hopscotch
- Knapsack - Counting Items
  - UVa 1213 - Sum of Different Primes
  - UVa 1158 - CubesSquared
- Knapsack - 0/1
  - UVa 11658 - Best Coalitions
  - UVa 10930 - A-Sequence
- Ad hoc

- UVa 10444 - Multi-peg Towers of Hanoi
- UVa 11375 - Matches
- UVa 10703 - sqrt log sin
- UVa 1231 - ACORN
- UVa 1239 - Greatest K-Palindrome Substring

dynamic-programming/473.cpp

```cpp
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#define MAX 10005
using namespace std;

int S[MAX], T[MAX];
char skip;
int main() {
    int n,t,m,cases;
    cin >> cases;
    while(cases--) {
        cin >> n >> t >> m;
        memset(T, 0x3F, sizeof(T));

        for(int i=1; i<=n; i++) {
            cin >> S[i];
            if (i<n) cin >> skip;
        }

        int k=0;
        T[0] = 0;

        for(int i=1;i<=n;i++) {
            for(int j=k; j>=0; j--) {
                int add = 0;
                if ((T[j]%t)+S[i] > t) add = t-T[j]%t;
                if (T[j]+S[i]+add <= T[j+1]) {
                    T[j+1] = T[j]+S[i]+add;
                    k=max(k, j+1);
                }
            }
        }

        int answer = 0;
        for(int i=0; i<=k && T[i] <= m*t; i++)
            answer = i;

        cout << answer << endl;
        if (cases) cout << endl;
    }

    return 0;
}
```

dynamic-programming/907.cpp

```cpp
#define MAX 602
#include <iostream>
#include <cstring>
#include <climits>
using namespace std;

int T[MAX][MAX], S[MAX], n, k;

int main() {
    while(cin >> n >> k) {
        n++; k++;
        memset(T, 0, sizeof(T));

        S[0] = 0;
        for(int i=1; i<=n; i++) {
            cin >> S[i];
        }
```

```
18
19          for(int i=1; i<=n; i++)
20              T[i][1] = T[i-1][1]+S[i];
21
22          for(int i=1; i<=k; i++)
23              T[1][i] = S[1];
24
25          for(int i=2; i<=n; i++) {
26              for(int j=2; j<=k; j++) {
27                  T[i][j] = INT_MAX;
28                  for(int x=1; x<i; x++)
29                      T[i][j] = min(T[i][j], max(T[x][j-1], T[i][1] - T[x][1]));
30              }
31          }
32
33          cout << T[n][k] << endl;
34      }
35  }
```

dynamic-programming/1158.cpp

```
1   #include <iostream>
2   #include <cstring>
3   #include <vector>
4   using namespace std;
5
6   int K[400001];
7   vector<int> W;
8
9   int main() {
10
11      for(int i=1; i*i*i<=400000; i++)
12          W.push_back(i*i*i);
13
14      for(int a=1, i=1; a<=400000; i++, a+=i*i)
15          W.push_back(a);
16
17      memset(K, 0x3f, sizeof(K));
18      K[0] = 0;
19      for(int i=0; i<W.size(); i++)
20          for(int j=W[i]; j<=400000; j++)
21              K[j] = min(K[j], K[j-W[i]]+1);
22
23      int n;
24      while(cin >> n, n!=-1)
25          cout << K[n] << endl;
26
27      return 0;
28  }
```

dynamic-programming/1207.cpp

```
1   #include <iostream>
2   #include <string>
3   #include <cstring>
4   #include <cmath>
5   #define MAX 1005
6   using namespace std;
7
8   int T[MAX][MAX];
9   string P, Q;
10
11
12  int main() {
13      int p, q;
14      while(cin >> p >> P >> q >> Q) {
```

```
        for(int i=0; i<=p; i++) { T[i][0] = i; }
        for(int i=0; i<=q; i++) { T[0][i] = i; }

        for(int i=1; i<=p; i++) {
            for(int j=1; j<=q; j++) {
                if (P[i-1] == Q[j-1])
                    T[i][j] = T[i-1][j-1];
                else
                    T[i][j] = min(min(T[i-1][j], T[i][j-1]), T[i-1][j-1])+1;
            }
        }

        cout << T[p][q] << endl;
    }

    return 0;
}
```

dynamic-programming/1213.cpp

```
#include <iostream>
#include <vector>
#include <cstring>
using namespace std;

long K[20][1300];
bool P[1300];
vector<int> W();

int main() {
    int n, k;

    memset(P, true, sizeof(P));
    P[0] = P[1] = false;
    for(int i=2; i<1300; i++) {
        if (P[i]) {
            W.push_back(i);
            for(int j=i*i; j<1300; j+=i)
                P[j] = false;
        }
    }

    K[0][0] = 1;
    for(int i=0; i<W.size(); i++)
        for(int p=19; p>0; p--)
            for(int j = W[i]; j<1300; j++)
                K[p][j]+=K[p-1][j-W[i]];

    while(cin >> n >> k, n|k)
        cout << K[k][n] << endl;
}
```

dynamic-programming/1223.cpp

```
#include <iostream>
#include <string>
#include <cstring>
#define MAX 5001
using namespace std;

int T[MAX][MAX];

int main() {
    int t; cin >> t; t=0;
    string s;
    while(cin >> s) {
```

```
        int sz = s.size();
        int maxx = 0;
        for(int i=1; i<=sz; i++) {
            for(int j=1; j<=sz; j++) {
                if (s[i-1] == s[j-1] && i!=j)
                    maxx = max(maxx, T[i][j] = T[i-1][j-1]+1);
                else
                    T[i][j] = 0;
            }
        }

        cout << maxx << endl;
    }

    return 0;
}
```

dynamic-programming/1231.cpp

```
#include <iostream>
#include <string>
#include <cstring>
#define MAX 2001
using namespace std;

int S[MAX][MAX], T[MAX][MAX], M[MAX];
char skip;
int main() {
    int cases; cin >> cases;
    while(cases--) {
        memset(S, 0, sizeof(S));
        memset(M, 0, sizeof(M));
        int t, h, f;
        cin >> t >> h >> f;

        for(int i=0; i<t; i++) {
            int k, a; cin >> k;
            while(k--) {
                cin >> a;
                S[a][i]++;
            }
        }

        for(int i=h; i>=0; i--) {
            for(int j=0; j<t; j++) {
                int move = i+f<=h ? M[i+f] : 0;
                int stay = i+1<=h ? T[i+1][j] : 0;
                T[i][j] = max(move, stay) + S[i][j];
                M[i] = max(M[i], T[i][j]);
            }
        }

        cout << M[0] << endl;
    }

    return 0;
}
```

dynamic-programming/1239.cpp

```
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#define MAX 1005
using namespace std;
```

```
 7
 8  int T[MAX][MAX];
 9
10  int main() {
11      int t; cin >> t; t=0;
12      string P;
13      int k;
14      while(cin >> P >> k) {
15          int p = P.size();
16
17          int maxx=0;
18          for(int i=p; i>=1; i--) {
19              for(int j=i; j<=p; j++) {
20                  T[i][j] = T[i+1][j-1] + (P[i-1] == P[j-1] ? 0 : 1);
21
22                  if (T[i][j] <= k)
23                      maxx = max(maxx, j-i+1);
24              }
25          }
26
27          cout << maxx << endl;
28      }
29
30      return 0;
31  }
```

dynamic-programming/10003.cpp

```
 1  #define MAX 1001
 2  #include <iostream>
 3  #include <cstring>
 4  #include <climits>
 5  using namespace std;
 6
 7  int T[MAX][MAX], S[MAX], n;
 8  bool V[MAX][MAX];
 9
10  int TT(int a, int b) {
11      if (a+1==b) return 0;
12      if (V[a][b]) return T[a][b];
13
14      int minn = INT_MAX;
15      for(int i=a+1; i<b; i++)
16          minn = min(minn, TT(a,i) + TT(i,b) + S[b]-S[a]);
17
18      V[a][b] = true;
19      return T[a][b] = minn;
20  }
21
22  int main() {
23      int t;
24      while(cin >> t, t) {
25          cin >> n;
26
27          memset(S, 0, sizeof(S));
28          memset(V, 0, sizeof(V));
29          S[0] = 0;
30          for(int i=1; i<=n; i++) {
31              cin >> S[i];
32          }
33          S[n+1] = t;
34
35          cout << "The minimum cutting is " << TT(0, n+1) << "." << endl;
36
37      }
38  }
```

dynamic-programming/10051.cpp

```cpp
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#include <climits>
#define MAX 501
#define MAXC 101
using namespace std;

int T[MAX][MAXC], F[MAX][MAXC], P[MAX][MAXC];
int A[6];

string translate(int side) {
    switch(side) {
        case 0: return "front";
        case 1: return "back";
        case 2: return "left";
        case 3: return "right";
        case 4: return "top";
        case 5: return "bottom";
    }
}

void print(int first, int k) {
    if (k==0) return;

    print(F[k][first], k-1);
    cout << T[k][first] << " " << translate(P[k][first]) << endl;
}

int main() {
    int n, n2, t;
    while(cin >> n, n) {
        if (t++) cout << endl;
        cout << "Case #" << t << endl;

        memset(T, 0, sizeof(T));

        for(int i=1;i<=MAXC; i++) {
            T[0][i] = 1;
        }
        int k = 0;

        for(int cube=1;cube<=n;cube++) {
            for(int i=0;i<6;i++) cin >> A[i];
            int newk = k;
            for(int j=k; j>=0; j--) {
                for(int i=0;i<6;i++) {
                    int other = (i/2*2)+(1-i%2);
                    if (T[j][A[i]] && !T[j+1][A[other]]) {
                        T[j+1][A[other]] = cube;
                        F[j+1][A[other]] = A[i];
                        P[j+1][A[other]] = i;
                        newk = max(newk, j+1);
                    }
                }
            }
            k=newk;
        }

        cout << k << endl;

        int first=0;
        for(int i=1;i<=100;i++)
            if (T[k][i]) first=i;

```

```
67          print(first, k);
68      }
69
70      return 0;
71  }
```

dynamic-programming/10066.cpp

```
 1  #include <iostream>
 2  #include <string>
 3  #include <cstring>
 4  #include <cmath>
 5  #define MAX 105
 6  using namespace std;
 7
 8  int T[MAX][MAX];
 9  int P[MAX], Q[MAX];
10
11  int main() {
12      int p, q, tt=0;
13      while(cin >> p >> q, tt++, p&&q) {
14          memset(T, 0, sizeof(T));
15
16          for(int i=0; i<p;i++) cin >> P[i];
17          for(int i=0; i<q;i++) cin >> Q[i];
18
19          for(int i=0; i<=p; i++) T[i][0] = 0;
20          for(int i=0; i<=q; i++) T[0][i] = 0;
21
22          for(int i=1; i<=p; i++) {
23              for(int j=1; j<=q; j++) {
24                  if (P[i-1] == Q[j-1])
25                      T[i][j] = T[i-1][j-1] + 1;
26                  else
27                      T[i][j] = max(T[i-1][j], T[i][j-1]);
28              }
29          }
30          cout << "Twin Towers #" << tt << endl;
31          cout << "Number of Tiles : " << T[p][q] << endl;
32          cout << endl;
33      }
34
35      return 0;
36  }
```

dynamic-programming/10154.cpp

```
 1  #include <iostream>
 2  #include <string>
 3  #include <cstring>
 4  #include <cmath>
 5  #include <climits>
 6  #include <vector>
 7  #include <algorithm>
 8  #define MAX 10005
 9  using namespace std;
10
11  struct Turtle {
12      int w,c;
13      Turtle() {}
14      Turtle(int w, int c) : w(w), c(c) {}
15  };
16
17  bool compare(const Turtle& a, const Turtle& b) {
18      return a.c > b.c;
19  }
```

```

vector<Turtle> V;
int T[MAX];
int main() {
    int w, c, k=0;
    T[0] = INT_MAX;

    while(cin >> w >> c) {
        V.push_back(Turtle(w, c-w));
    }
    sort(V.begin(), V.end(), compare);

    for(int i=0; i<V.size(); i++) {
        int w = V[i].w, c = V[i].c;

        for(int j=k; j>=0; j--) {
            int next = min(T[j]-w, c);
            if (next >= T[j+1]) {
                T[j+1] = next;
                k=max(k, j+1);
            }
        }
    }
    cout << k << endl;

    return 0;
}
```

dynamic-programming/10192.cpp

```
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#define MAX 1005
using namespace std;

int T[MAX][MAX];
string P, Q;

int main() {
    int p, q, tt=0;
    while(getline(cin, P), P!="#") {
        tt++;
        getline(cin, Q);
        int p = P.size(), q = Q.size();

        memset(T, 0, sizeof(T));

        for(int i=0; i<=p; i++) T[i][0] = 0;
        for(int i=0; i<=q; i++) T[0][i] = 0;

        for(int i=1; i<=p; i++) {
            for(int j=1; j<=q; j++) {
                if (P[i-1] == Q[j-1])
                    T[i][j] = T[i-1][j-1] + 1;
                else
                    T[i][j] = max(T[i-1][j], T[i][j-1]);
            }
        }
        cout << "Case #" << tt << ": you can visit at most " << T[p][q] << " cities." << endl;
    }

    return 0;
}
```

dynamic-programming/10259.cpp

```cpp
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#include <climits>
#define MAX 101
using namespace std;

int T[MAX][MAX], M[MAX][MAX], n, k;

int walk(int x, int y, int curr) {
    if (x < 0 || x >= n || y < 0 || y >= n) return 0;
    if (T[x][y] <= curr) return 0;
    if (M[x][y] >= 0) return M[x][y];

    int maxx = 0;
    for(int i=1; i<=k; i++) {
        maxx = max(maxx, walk(x-i, y, T[x][y])+T[x][y]);
        maxx = max(maxx, walk(x+i, y, T[x][y])+T[x][y]);
        maxx = max(maxx, walk(x, y-i, T[x][y])+T[x][y]);
        maxx = max(maxx, walk(x, y+i, T[x][y])+T[x][y]);
    }
    return M[x][y] = maxx;
}

int main() {
    int t;
    cin >> t;
    while(t--) {
        cin >> n >> k;
        memset(M, -1, sizeof(M));
        for(int i=0; i<n; i++)
            for(int j=0; j<n; j++)
                cin >> T[i][j];

        cout << walk(0,0, -1) << endl;
        if (t) cout << endl;
    }

    return 0;
}
```

dynamic-programming/10304.cpp

```cpp
#define MAX 252
#include <iostream>
#include <cstring>
#include <climits>
using namespace std;

int T[MAX][MAX], S[MAX], n;
bool V[MAX][MAX];

int TT(int a, int b) {
    if (b < a) return 0;
    if (V[a][b]) return T[a][b];

    int minn = INT_MAX;
    for(int i=a; i<=b; i++)
        minn = min(minn, TT(a,i-1) + TT(i+1,b) + (S[b]-S[a-1])-(S[i]-S[i-1]));

    V[a][b] = true;
    return T[a][b] = minn;
}

int main() {
    while(cin >> n) {
        memset(V, 0, sizeof(V));
```

```
25          S[0] = 0;
26          for(int i=1; i<=n; i++) {
27              cin >> S[i];
28              S[i] += S[i-1];
29          }
30
31          cout << TT(1, n) << endl;
32
33      }
34  }
```

dynamic-programming/10444.cpp

```
1   #include <iostream>
2   #include <string>
3   #include <cstring>
4   #include <cmath>
5   #include <climits>
6   #define MAX 205
7   using namespace std;
8
9   int T[MAX][MAX];
10  int main() {
11      int n=201, p=21, t=0;
12
13      for(int i=0;i<=n;i++) {
14          if (i<31)
15              T[i][3] = (1<<i)-1;
16          else
17              T[i][3] = INT_MAX; //avoid overflow
18      }
19
20      for(int i=1; i<=n; i++) {
21          for(int j=4; j<=p; j++) {
22              if (i<j) {
23                  T[i][j] = 2*i-1;
24              } else {
25                  int minn = INT_MAX;
26                  for(int k=1;k<i;k++) {
27                      int value = 2*T[k][j]+T[i-k][j-1];
28                      if (value >= 0) //avoid overflow
29                          minn = min(minn, value);
30                  }
31                  T[i][j] = minn;
32              }
33
34          }
35      }
36
37      while(cin >> n >> p, n | p) {
38          cout << "Case " << ++t << ": " << T[n][p] << endl;
39      }
40
41      return 0;
42  }
```

dynamic-programming/10635.cpp

```
1   #include <iostream>
2   #include <cstring>
3   #include <climits>
4   #include <cmath>
5   #include <algorithm>
6   #define MAX 255*255
7   using namespace std;
8
```

```
 9  int P[MAX], Q[MAX], M[MAX];
10
11  int main() {
12      int n, p, q, tt=0, temp;
13      cin >> n;
14      while(cin >> n >> p >> q) {
15          memset(P, 0, sizeof(P));
16          q++; p++;
17          for(int i=1;i<=p;i++) {
18              cin >> temp;
19              P[temp] = i;
20          }
21
22          for (int i=1;i<=q;i++) {
23              cin >> temp;
24              Q[i] = P[temp];
25          }
26
27          int k=0; M[0]=0;
28          for(int i=1;i<=q;i++) {
29              if (Q[i] > M[k]) {
30                  k++; M[k] = Q[i];
31              } else {
32                  int j = (int)(lower_bound(M, M+k+1, Q[i])-M);
33                  if (Q[i] > M[j]) j++;
34                  M[j] = Q[i];
35              }
36          }
37
38          cout << "Case " << ++tt << ": " << k << endl;
39      }
40
41      return 0;
42  }
```

dynamic-programming/10684.cpp

```
 1  #include <iostream>
 2  #include <cmath>
 3  #define MAX 1005
 4  using namespace std;
 5
 6  int main() {
 7      int n, a;
 8      while(cin >> n, n) {
 9          int t=0, s=0;
10          for(int i=0;i<n;i++) {
11              cin >> a;
12              if (s+a>=0)
13                  t = max(t, s+=a);
14              else
15                  s = 0;
16          }
17          if (s>0) {
18              cout << "The maximum winning streak is " << t << "." << endl;
19          } else {
20              cout << "Losing streak." << endl;
21          }
22      }
23
24      return 0;
25  }
```

dynamic-programming/10723.cpp

```
 1  #include <iostream>
```

```
#include <string>
#include <cstring>
#include <cmath>
#define MAX 1005
using namespace std;

int T[MAX][MAX], D[MAX][MAX];
string P, Q;

int main() {
    int p, q, t, tt=0;
    cin >> t;
    getline(cin, P);
    while(tt++ < t) {
        getline(cin, P);
        getline(cin, Q);
        int p = P.size(), q = Q.size();

        for(int i=0; i<=p; i++) { T[i][0] = 0; D[i][0] = 1; }
        for(int i=0; i<=q; i++) { T[0][i] = 0; D[0][i] = 1; }

        for(int i=1; i<=p; i++) {
            for(int j=1; j<=q; j++) {
                D[i][j] = 0;
                if (P[i-1] == Q[j-1]) {
                    T[i][j] = T[i-1][j-1] + 1;
                    D[i][j] = D[i-1][j-1];
                }
                else {
                    T[i][j] = max(T[i-1][j], T[i][j-1]);
                    if (T[i-1][j] == T[i][j]) D[i][j] += D[i-1][j];
                    if (T[i][j-1] == T[i][j]) D[i][j] += D[i][j-1];
                }
            }
        }

        cout << "Case #" << tt << ": " << p+q-T[p][q] << " " << D[p][q] << endl;
    }

    return 0;
}
```

dynamic-programming/10739.cpp

```
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#define MAX 1005
using namespace std;

int T[MAX][MAX];
string P, Q;

int main() {
    int p, q, t, tt=0;
    cin >> t;
    getline(cin, P);
    while(tt++ < t) {
        getline(cin, P);
        Q = string(P.rbegin(), P.rend());
        int p = P.size(), q = Q.size();

        for(int i=0; i<=p; i++) { T[i][0] = i; }
        for(int i=0; i<=q; i++) { T[0][i] = i; }

        for(int i=1; i<=p; i++) {
            for(int j=1; j<=q; j++) {
```

```
                if (P[i-1] == Q[j-1])
                    T[i][j] = T[i-1][j-1];
                else
                    T[i][j] = min(min(T[i-1][j], T[i][j-1]), T[i-1][j-1])+1;
            }
        }

        cout << "Case " << tt << ": " << T[p][q]/2 << endl;
    }

    return 0;
}
```

dynamic-programming/10827.cpp

```
#include <iostream>
#include <climits>
#define MAX 160
using namespace std;

int T[MAX][MAX];

int main() {
    int n, a, cases;
    cin >> cases;
    while(cin >> n) {
        for(int i=1; i<=n; i++) {
            for(int j=1;j<=n;j++) {
                cin >> T[i][j];
                T[i+n][j] = T[i][j];
            }
        }

        for(int i=1; i<=2*n; i++)
            for(int j=1;j<=n; j++)
                T[i][j]+=T[i-1][j];

        int t = 0;
        for(int i=1;i<=2*n; i++) {
            for(int j=i;j<=min(i+n-1, 2*n);j++) {
                int smax=0, smin=0, ssum=0, tmax=0, tmin=0;
                for(int k=1;k<=n; k++)
                    ssum += T[j][k] - T[i-1][k];

                for(int k=1;k<=n; k++) {
                    int a = T[j][k] - T[i-1][k];
                    smax += a;
                    smin += a;

                    tmax = max(tmax, smax);
                    tmin = min(tmin, smin);

                    if (smax < 0) smax = 0;
                    if (smin > 0) smin = 0;
                }
                t = max(t, max(tmax, ssum-tmin));
            }
        }

        cout << t << endl;
    }

    return 0;
}
```

dynamic-programming/10891.cpp

```cpp
#define MAX 101
#include <iostream>
#include <cstring>
#include <climits>
using namespace std;

int T[MAX][MAX], S[MAX], n;
bool V[MAX][MAX];

int TT(int a, int b) {
    if (b<a) return 0;
    if (V[a][b]) return T[a][b];

    int maxx = INT_MIN;
    for(int i=a; i<=b; i++)
        maxx = max(maxx, S[b]-S[a-1] - TT(i+1,b));

    for(int i=b; i>=a; i--)
        maxx = max(maxx, S[b]-S[a-1] - TT(a,i-1));

    V[a][b] = true;
    return T[a][b] = maxx;
}

int main() {
    while(cin >> n, n) {
        memset(S, 0, sizeof(S));
        memset(V, 0, sizeof(V));
        S[0] = 0;
        for(int i=1; i<=n; i++) {
            cin >> S[i];
            S[i] += S[i-1];
        }

        cout << 2*TT(1, n)-S[n]-S[0] << endl;

    }
}
```

dynamic-programming/10930.cpp

```cpp
#include <iostream>
#include <cstring>
#include <iomanip>
using namespace std;

int K[30001];

int main() {
    int n, t=0, w;
    while(t++, cin >> n) {
        cout << "Case #" << t << ":";
        memset(K, 0, sizeof(K));

        bool ok=true;

        K[0] = 1; int last = 0;
        for(int i=1; i<=n; i++) {
            cin >> w; cout << " " << w;
            ok &= !K[w] && w > last;
            for(int j=10000; j>=w; j--)
                if (K[j-w])
                    K[j] = 1;
            last = w;
        }
        cout << endl;
        cout << "This is" << (ok?"":" not") << " an A-sequence." << endl;
    }
```

```
28
29      return 0;
30  }
```

dynamic-programming/11003.cpp

```
 1  #include <iostream>
 2  #include <string>
 3  #include <cstring>
 4  #include <cmath>
 5  #include <climits>
 6  #define MAX 10005
 7  using namespace std;
 8
 9  int T[MAX];
10  int main() {
11      int n, w, c;
12      while(cin >> n, n) {
13          memset(T, 0, sizeof(T));
14
15          int k = 0;
16          T[0] = INT_MAX;
17          for(int i=1; i<=n; i++) {
18              cin >> w >> c;
19              for(int j=k; j>=0; j--) {
20                  int next = min(T[j]-w, c);
21                  if (next >= T[j+1]) {
22                      T[j+1] = next;
23                      k=max(k, j+1);
24                  }
25              }
26          }
27
28          cout << k << endl;
29      }
30
31      return 0;
32  }
```

dynamic-programming/11059.cpp

```
 1  #include <iostream>
 2  #include <climits>
 3  #include <cmath>
 4  #define MAX 1005
 5  using namespace std;
 6
 7  int main() {
 8      long long n, a, t=0;
 9      while(cin >> n) {
10          long long maxx=0, newneg=0, newpos=0, spos=1, sneg=1;
11          bool valid = false;
12          for(int i=0;i<n;i++) {
13              cin >> a;
14              if (spos*a>0) {
15                  valid = true;
16                  spos*=a;
17              } else {
18                  newneg = spos*a;
19                  spos = 1;
20              }
21
22
23              if (sneg*a<0) {
24                  sneg*=a;
25              } else  {
```

```
                if (sneg*a>0) valid = true;
                newpos = sneg*a;
                sneg = 1;
            }

            maxx = max(maxx, spos = max(spos, newpos));
            sneg = min(sneg, newneg);
            newpos = newneg = 0;
        }
        if (!valid) maxx = 0;
        cout << "Case #" << ++t << ": The maximum product is " << maxx << "." << endl;
        cout << endl;
    }

    return 0;
}
```

dynamic-programming/11151.cpp

```
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#define MAX 1005
using namespace std;

int T[MAX][MAX];
string P, Q;

int main() {
    int p, q, t;
    cin >> t;
    getline(cin, P);
    while(t--) {
        getline(cin, P);
        Q = string(P.rbegin(), P.rend());
        int p = P.size(), q = Q.size();

        for(int i=0; i<=p; i++) { T[i][0] = 0;  }
        for(int i=0; i<=q; i++) { T[0][i] = 0;  }

        for(int i=1; i<=p; i++) {
            for(int j=1; j<=q; j++) {
                if (P[i-1] == Q[j-1]) {
                    T[i][j] = T[i-1][j-1] + 1;
                }
                else {
                    T[i][j] = max(T[i-1][j], T[i][j-1]);
                }
            }
        }

        cout << T[p][q] << endl;
    }

    return 0;
}
```

dynamic-programming/11375.cpp

```
#include <iostream>
#include <vector>
#include <cstring>
using namespace std;

int K[] = {6, 2, 5, 5, 4, 5, 6, 3, 7, 6};
```

```
vector<int> T[2001][10];

void add(vector<int> &a, const vector<int> &b) {
    int carry = 0;
    for(int i=0;i<max(a.size(), b.size());i++) {
        int aa = i<a.size()?a[i]:0;
        int bb = i<b.size()?b[i]:0;
        int cc = aa+bb+carry;
        if (i >= a.size()) a.push_back(0);
        a[i] = cc%10;
        carry = cc/10;
    }
    if (carry)
        a.push_back(carry);
}

int main() {
    vector<int> one; one.push_back(1);

    for(int i=2; i<2001; i++) {
        for(int j=0;j<10; j++)
            if (i>=K[j]) {
                add(T[i][j], one);
                for(int k=0;k<10;k++)
                    add(T[i][j], T[i-K[j]][k]);
            }
    }

    int n;
    while(cin >> n) {
        vector<int> ans = n>=6?one:vector<int>();
        for(int i=1;i<10;i++)
            add(ans, T[n][i]);

        for(int i=ans.size()-1;i>=0;i--) {
            cout << ans[i];
        }
        if (ans.size()==0) cout << 0;
        cout << endl;
    }

    return 0;
}
```

dynamic-programming/11658.cpp

```
#include <iostream>
#include <cstring>
#include <iomanip>
using namespace std;

int K[10001], W[102];

int main() {
  int n, x, a, b;

  while(cin >> n >> x, n|x) {
    memset(K, 0, sizeof(K));

    for(int i=1; i<=n; i++) {
        cin >> a; cin.ignore(); cin >> b;
        W[i] = a*100+b;
    }

    K[W[x]] = 1;
    for(int i=1; i<=n; i++) {
        if (i==x) continue;
        for(int j=10000; j>=W[i]; j--)
```

```
            if (K[j-W[i]])
                K[j] = 1;
    }

    int maxx = 0;
    for(int i=5001; i<=10000; i++) {
        if (K[i]) {
            maxx = i;
            break;
        }
    }

    cout << fixed << setprecision(2) << (W[x]/((double)maxx)*100.0) << endl;
  }

  return 0;
}
```

dynamic-programming/11703.cpp

```
#include <iostream>
#include <cmath>
#include <cstring>
#include <cassert>
using namespace std;

int K[1000001];

int main() {
    K[0] = 1;
    for(int i=1; i<1000001; i++) {
        int a = (int)(i-sqrt(i));
        int b = (int)log(i);
        int c = (int)(i*pow(sin(i), 2));
        K[i] = (K[a] + K[b] + K[c])%1000000;
    }

    int n;
    while(cin >> n, n>-1)
        cout << K[n] << endl;

    return 0;
}
```

dynamic-programming/12147.cpp

```
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#define MAX 1005
using namespace std;

int T[MAX][MAX];
int S[MAX][MAX];
string P, Q;

int main() {
    int k;
    while(cin >> k, k) {
        cin >> P >> Q;
        int p = P.size(), q = Q.size();

        for(int i=0; i<=p; i++) T[i][0] = S[i][0] = 0;
        for(int i=0; i<=q; i++) T[0][i] = S[0][i] = 0;

```

```
21          for(int i=1; i<=p; i++) {
22              for(int j=1; j<=q; j++) {
23                  if (P[i-1] == Q[j-1])
24                      S[i][j] = S[i-1][j-1] + 1;
25                  else
26                      S[i][j] = 0;
27              }
28          }
29
30          for(int i=1; i<=p; i++) {
31              for(int j=1; j<=q; j++) {
32                  T[i][j] = max(T[i-1][j], T[i][j-1]);
33
34                  for(int s=k; s<=S[i][j]; s++)
35                      T[i][j] = max(T[i][j], T[i-s][j-s]+s);
36              }
37          }
38          cout << T[p][q] << endl;
39      }
40
41      return 0;
42  }
```

Graphs

- DFS
  - UVa 10243 - Fire! Fire! Fire!
  - UVa 1220 - Party at Hali-Bula
  - UVa 12186 - Another Crisis
  - UVa 273 - Jack Straws
  - UVa 1216 - The Bug Sensor Problem
  - UVa 1197 - The Suspects
- Flood Fill
  - UVa 11110 - Equidivisions
  - UVa 11518 - Dominos 2
- Shortest Path (BFS)
  - UVa 314 - Robot
  - UVa 321 - The New Villa
  - UVa 627 - The Net
  - UVa 12101 - Prime Path
  - UVa 10044 - Erdos Number
  - UVa 12260 - Unlock the Lock
  - UVa 12135 - Switch Bulbs
  - UVa 298 - Race Tracks
- Maximum Flow (Ford-Fulkerson with DFS)
  - UVa 820 - Internet Bandwidth
  - UVa 10480 - Sabotage
  - UVa 10092 - The Problem with the Problem Setter
  - UVa 10511 - Councilling
- Bipartite Matching/Konig Theorem
  - UVa 11419 - SAM I AM
  - UVa 12168 - Cat vs. Dog
  - UVa 12159 - Gun Fight
- Minimum Spanning Tree (Prim)
  - UVa 1235 - Anti Brute Force Lock
  - UVa 1208 - Oreon
  - UVa 908 - Re-connecting Computer Sites
- Minimum Spanning Tree (Prim with Priority Queue)
  - UVa 11631 - Dark roads
  - UVa 11733 - Airports
  - UVa 11747 - Heavy Cycle Edges
  - UVa 10397 - Connect the Campus
  - UVa 1234 - RACING
  - UVa 1174 - IP-TV
- Strongly Connected Components
  - UVa 11838 - Come and Go
  - UVa 11709 - Trust Groups
  - UVa 1223 - Sub-Dictionary
- Finding Articulation Points/Bridges
  - UVa 315 - Network

    - UVa 769 - Critical Links

- Topological Sorting
    - UVa 11686 - Pick up Sticks
    - UVa 1263 - Mines
    - UVa 11770 - Lighting Away

- Minimum Path (Floyd-Warshall)
    - UVa 10278 - Fire Station
    - UVa 12173 - Randomly-priced Tickets
    - UVa 1056 - Degrees of Separation

- Minimum Path (Dijkstra)
    - UVa 10986 - Sending email
    - UVa 10389 - Subway
    - UVa 11833 - Route Change
    - UVa 12144 - Almost Shortest Path
    - UVa 1247 - Interstar Transport

graphs/273.cpp

```cpp
#include <iostream>
#include <cstring>
#define MAX 100002
using namespace std;

static bool segment(int xi, int yi, int xj, int yj,
                    int xk, int yk) {
  return (xi <= xk || xj <= xk) && (xk <= xi || xk <= xj) &&
         (yi <= yk || yj <= yk) && (yk <= yi || yk <= yj);
}

static char direction(int xi, int yi, int xj, int yj,
                          int xk, int yk) {
  int a = (xk - xi) * (yj - yi);
  int b = (xj - xi) * (yk - yi);
  return a < b ? -1 : a > b ? 1 : 0;
}

bool intersect(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4) {
  char d1 = direction(x3, y3, x4, y4, x1, y1);
  char d2 = direction(x3, y3, x4, y4, x2, y2);
  char d3 = direction(x1, y1, x2, y2, x3, y3);
  char d4 = direction(x1, y1, x2, y2, x4, y4);
  return (((d1 > 0 && d2 < 0) || (d1 < 0 && d2 > 0)) &&
          ((d3 > 0 && d4 < 0) || (d3 < 0 && d4 > 0))) ||
         (d1 == 0 && segment(x3, y3, x4, y4, x1, y1)) ||
         (d2 == 0 && segment(x3, y3, x4, y4, x2, y2)) ||
         (d3 == 0 && segment(x1, y1, x2, y2, x3, y3)) ||
         (d4 == 0 && segment(x1, y1, x2, y2, x4, y4));
}

int G[20][20], V[20], A[20], B[20], C[20], D[20], n;

int dfs(int v, int comp) {
    V[v] = comp;
    for(int i=1; i<=n; i++)
        if (!V[i] && G[v][i])
            dfs(i, comp);
}

int main() {
    int t; cin >> t; t=0;
    while(cin >> n) {
        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));
        for(int i=1; i<=n; i++) {
            cin >> A[i] >> B[i] >> C[i] >> D[i];
            for(int j=1;j<i; j++)
                G[i][j] = G[j][i] = intersect(A[i], B[i], C[i], D[i], A[j], B[j], C[j], D[j]);
        }

        int compn = 0;
        for(int i=1; i<=n; i++)
            if (!V[i])
                dfs(i, ++compn);

        if (t++) cout << endl;
        int a, b;
        while(cin >> a >> b, a|b) {
            cout << (V[a] == V[b]?"CONNECTED":"NOT CONNECTED") << endl;
        }
    }
    return 0;
}
```

graphs/298.cpp

```cpp
#include <iostream>
#include <cstring>
#include <queue>
#define MAX 30
using namespace std;

bool V[MAX][MAX][7][7];
int X, Y;

struct Step {
    int x, y, a, b, v;
    Step() {}
    Step(int x, int y, int a, int b, int v) : x(x), y(y), a(a), b(b), v(v) {}

    bool valid() {
        return x>=0 && x<X && y>=0 && y<Y && a >= -3 && a <= 3 && b >= -3 && b <= 3 && !V[x][y][a+
    }

    void mark() {
        V[x][y][a+3][b+3] = true;
    }

    Step go(int mx, int my) {
        return Step(x+a+mx, y+b+my, a+mx, b+my, v+1);
    }
};

int main() {
    int t; cin >> t; t=0;
    while(cin >> X >> Y) {
        memset(V, 0, sizeof(V));
        int x1, y1, x2, y2;
        cin >> x1 >> y1 >> x2 >> y2;

        int p, px1, px2, py1, py2;
        cin >> p;
        while(p--) {
            cin >> px1 >> px2 >> py1 >> py2;
            for(int i=px1; i<=px2; i++)
                for(int j=py1; j<=py2; j++)
                    for(int ai=0; ai<=6; ai++)
                        for(int bi=0; bi<=6; bi++)
                            V[i][j][ai][bi] = true;
        }

        bool found = false;
        queue<Step> Q;
        Q.push(Step(x1, y1, 0, 0, 0));

        while(!Q.empty()) {
            Step it = Q.front(); Q.pop();
            if (!it.valid()) continue;
            it.mark();

            if (it.x == x2 && it.y == y2) {
                cout << "Optimal solution takes " << it.v << " hops." << endl;
                found = true;
                break;
            }

            for(int ai=-1; ai<=1; ai++)
                for(int bi=-1; bi<=1; bi++)
                    Q.push(it.go(ai, bi));
        }
        if (!found) cout << "No solution." << endl;
    }
```

```
}
```

graphs/314.cpp

```
#include <iostream>
#include <iomanip>
#include <cstring>
#include <string>
#include <cmath>
#include <climits>
#include <vector>
#define MAX 70
using namespace std;

int G[MAX][MAX], n, m, sx, sy, tx, ty;
bool V[MAX][MAX][4];
string dir;

struct Step {
    int x, y, d, v, p;
    Step() {}
    Step(int x, int y, int d, int v, int p) : x(x), y(y), d(d), v(v), p(p) {}
    Step left(int pp) {
        return Step(x, y, (d+3)%4, v+1, pp);
    }
    Step right(int pp) {
        return Step(x, y, (d+1)%4, v+1, pp);
    }
    bool canGo(int i) {
        return (d==0 && x-i>=1 && !G[x-i][y]) ||
               (d==1 && y+i<m-1 && !G[x][y+i]) ||
               (d==2 && x+i<n-1 && !G[x+i][y]) ||
               (d==3 && y-i>=1 && !G[x][y-i]);
    }
    Step go(int pp, int i) {
        if (d==0) return Step(x-i, y, d, v+1, pp);
        if (d==1) return Step(x, y+i, d, v+1, pp);
        if (d==2) return Step(x+i, y, d, v+1, pp);
        if (d==3) return Step(x, y-i, d, v+1, pp);
    }

};

int main() {
    while(cin >> n >> m, n|m) {
        vector<Step> Q;

        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));

        for(int i=0;i<n;i++)
            for (int j=0;j<m;j++)
                cin >> G[i][j];

        n++; m++;
        for(int i=n-1;i>=0;i--)
            for (int j=m-1;j>=0;j--)
                if (G[i][j])
                    G[i+1][j] = G[i][j+1] = G[i+1][j+1] = 1;

        cin >> sx >> sy >> tx >> ty >> dir;
        if (dir=="north") Q.push_back(Step(sx, sy, 0, 0, -1));
        if (dir=="east") Q.push_back(Step(sx, sy, 1, 0, -1));
        if (dir=="south") Q.push_back(Step(sx, sy, 2, 0, -1));
        if (dir=="west") Q.push_back(Step(sx, sy, 3, 0, -1));

        int ptr = 0;
        while(ptr < Q.size()) {
```

```
            Step it = Q[ptr];
            if (it.x == tx && it.y == ty) {
                cout << it.v << endl;
                break;
            }

            if (V[it.x][it.y][it.d]) { ptr++; continue; }
            V[it.x][it.y][it.d] = true;

            Q.push_back(it.left(ptr));
            Q.push_back(it.right(ptr));
            for (int i=1; i<=3 && it.canGo(i); i++)
                Q.push_back(it.go(ptr, i));

            ptr++;
        }
        if (ptr == Q.size()) cout << -1 << endl;
    }
}
```

graphs/315.cpp

```
#include <iostream>
#include <cstring>
#include <string>
#include <sstream>
#define MAX 101
using namespace std;
int G[MAX][MAX], V[MAX], L[MAX], P[MAX], n, gpe;

void dfs(int u, int v) {
    V[v] = L[v] = ++gpe;
    for(int i = 1; i <= n; i++) {
        if(G[v][i]) {
            if(!V[i]){
                dfs(v, i);
                L[v] = min(L[v], L[i]);
                if(L[i] >= V[v]) P[v]++;
            } else if(i != u) {
                L[v] = min(L[v], V[i]);
            }
        }
    }
}

int main() {
    while(cin >> n, n) {
        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));
        memset(L, 0, sizeof(L));
        memset(P, 0, sizeof(P));
        gpe = 0;

        int a, b; string s;
        while(getline(cin, s), s != "0") {
            stringstream sin(s);
            sin >> a;
            while(sin >> b) {
                G[a][b] = G[b][a] = 1;
            }
        }
        dfs(1, 1); P[1]--;
        int cnt = 0;
        for(int i=1; i<=n; i++)
            if (P[i]) cnt++;

        cout << cnt << endl;
    }
```

```
47  }
```

graphs/321.cpp

```
 1  #include <iostream>
 2  #include <cstring>
 3  #include <climits>
 4  #include <vector>
 5  #define MAX 15
 6  using namespace std;
 7
 8
 9
10  int G[MAX][MAX], C[MAX][MAX], n, m1, m2;
11  bool V[MAX][1200];
12  string dir;
13  struct Step {
14      int x, s, v, p;
15      int type, room;
16      Step() {}
17      Step(int x, int s, int v, int p) : x(x), s(s), v(v), p(p) {}
18      Step(int x, int s, int v, int p, int type, int room) : type(type), room(room), x(x), s(s), v(v
19
20      Step change(int pp, int i) {
21          return Step(x, s ^ (1<<i), v+1, pp, (s & (1<<i))?2:1, i);
22      }
23      Step move(int pp, int i) {
24          return Step(i, s, v+1, pp, 3, i);
25      }
26  };
27
28  vector<Step> Q;
29
30  void print(Step step) {
31          if (step.p == -1) return;
32          print(Q[step.p]);
33          if (step.type == 1)
34              cout << "- Switch on light in room " << step.room+1 << "." << endl;
35          if (step.type == 2)
36              cout << "- Switch off light in room " << step.room+1 << "." << endl;
37          if (step.type == 3)
38              cout << "- Move to room " << step.room+1 << "." << endl;
39      }
40
41
42
43  int main() {
44      int tt=0;
45      while(cin >> n >> m1 >> m2, n|m1|m2) {
46          Q = vector<Step>();
47
48          memset(G, 0, sizeof(G));
49          memset(C, 0, sizeof(C));
50          memset(V, 0, sizeof(V));
51
52          int a, b;
53          for(int i=0;i<m1; i++) {
54              cin >> a >> b;
55              a--; b--;
56              G[a][b] = G[b][a] = 1;
57          }
58          for(int i=0;i<m2; i++) {
59              cin >> a >> b;
60              a--;b--;
61              C[a][b] = 1;
62          }
63
64          Q.push_back(Step(0, 1, 0, -1));
```

```

        int ptr = 0;
        cout << "Villa #" << ++tt << endl;
        while(ptr < Q.size()) {
            Step it = Q[ptr];
            if (it.x == n-1 && it.s == (1<<(n-1))) {
                cout << "The problem can be solved in " << it.v << " steps:" << endl;
                print(it);
                break;
            }

            if (V[it.x][it.s]) { ptr++; continue; }
            V[it.x][it.s] = true;

            for(int i=0; i<n; i++) {
                if (G[it.x][i] && (it.s & (1<<i))) Q.push_back(it.move(ptr, i));
                if (C[it.x][i] && it.x != i) Q.push_back(it.change(ptr, i));
            }

            ptr++;
        }
        if (ptr == Q.size()) cout << "The problem cannot be solved." << endl;
        cout << endl;
    }
}
```

graphs/627.cpp

```
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#define MAX 400
using namespace std;

int G[MAX][MAX], n, m;
bool V[MAX];


struct Step {
    int x, v, p;
    Step() {}
    Step(int x, int v, int p) : x(x), v(v), p(p) {}
};

vector<Step> Q;

void print(Step step, bool first) {
    if (step.p != -1) print(Q[step.p], false);
    cout << step.x << (first?"":" ");
}

int main() {
    while(cin >> n) {
        cout << "-----" << endl;
        memset(G, 0, sizeof(G));

        int a, b;
        for(int i=0; i<n;i++) {
            cin >> a;
            while(cin.get()!='\n') {
                if (cin.peek() == '\n') break;
                cin >> b;
                G[a][b] = true;
            }
        }

        cin >> m;
```

```
        for(int i=0;i<m;i++) {
            memset(V, 0, sizeof(V));
            cin >> a >> b;
            Q = vector<Step>();
            Q.push_back(Step(a, 0, -1));

            int ptr = 0;
            while(ptr < Q.size()) {
                Step it = Q[ptr];
                if (it.x == b) {
                    print(it, true);
                    cout << endl;
                    break;
                }

                if (V[it.x]) { ptr++; continue; }
                V[it.x] = true;

                for(int i=1; i<=n; i++)
                    if (G[it.x][i]) Q.push_back(Step(i,it.v+1,ptr));

                ptr++;
            }
            if (ptr == Q.size()) cout << "connection impossible" << endl;
        }
    }
}
```

graphs/796.cpp

```
#include <iostream>
#include <cstring>
#include <string>
#include <sstream>
#include <vector>
#include <algorithm>
#define MAX 101
using namespace std;
int G[MAX][MAX], V[MAX], L[MAX], n, gpe;

struct Ponte {
    int a, b;
    Ponte() { }
    Ponte(int a, int b) : a(min(a, b)), b(max(a, b)) {}
};
bool comp(const Ponte& a, const Ponte& b) { return a.a < b.a || (a.a==b.a && a.b < b.b); }
vector<Ponte> P;

void dfs(int u, int v) {
    V[v] = L[v] = ++gpe;
    for(int i = 0; i < n; i++) {
        if(G[v][i]) {
            if(!V[i]){
                dfs(v, i);
                L[v] = min(L[v], L[i]);
                if(L[i] == V[i]) P.push_back(Ponte(v, i));
            } else if(i != u) {
                L[v] = min(L[v], V[i]);
            }
        }
    }
}

int main() {
    while(cin >> n) {
        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));
        memset(L, 0, sizeof(L));
```

```
        P.clear();
        gpe = 0;

        int a, an, b; char skip;
        for(int i=0; i<n; i++) {
            cin >> a >> skip >> an >> skip;
            while(an--) {
                cin >> b; G[b][a] = G[a][b] = 1;
            }
        }

        for(int i=0; i<n; i++)
            if (!V[i])
                dfs(i, i);

        cout << P.size() << " critical links" << endl;
        sort(P.begin(), P.end(), comp);

        for(int i=0;i<P.size(); i++) {
            cout << P[i].a << " - " << P[i].b << endl;
        }
        cout << endl;
    }
}
```

graphs/820.cpp

```
#include <iostream>
#include <iomanip>
#include <cstring>
#include <string>
#include <climits>
#include <cmath>
#define MAX 1006
using namespace std;

int G[MAX][MAX], n;
int F[MAX][MAX];
bool V[MAX];

int send(int s, int t, int minn) {
    V[s] = true;

    if (s==t) return minn;

    for(int i=1; i<=n; i++) {
        int capacity = G[s][i]-F[s][i];
        if (!V[i] && G[s][i]-F[s][i] > 0) {
            if (int sent = send(i, t, min(minn, capacity))) {
                F[s][i] += sent;
                F[i][s] -= sent;
                return sent;
            }
        }
    }

    return 0;
}

int main() {
    int tt=0;
    while(cin >> n, n) {
        memset(G, 0, sizeof(G));
        memset(F, 0, sizeof(F));
        memset(V, 0, sizeof(V));

        tt++;
        int s, t, c;
```

```
        cin >> s >> t >> c;
        for(int i=0;i<c;i++) {
            int a, b, f;
            cin >> a >> b >> f;
            G[a][b] = G[b][a] += f;
        }

        int total = 0;
        while(int sent = send(s, t, INT_MAX)) {
            total += sent;
            memset(V, 0, sizeof(V));
        }

        cout << "Network " << tt << endl;
        cout << "The bandwidth is " << total << "." <<  endl;
        cout << endl;
    }


}
```

graphs/908.cpp

```
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#define MAX 200010

struct Road {
    int v, c;
    Road(int v, int c) : v(v), c(c) {}
    bool operator < (const Road& that) const { return c > that.c; }
};

using namespace std;

vector<Road> G[MAX];

int main() {
    int n, k, m, t=0;
    while(cin >> n) {
        memset(G, 0, sizeof(G));

        int before = 0;
        for(int i=0; i<n-1; i++) {
            int a, b, c;
            cin >> a >> b >> c;
            G[a].push_back(Road(b, c));
            G[b].push_back(Road(a, c));
            before += c;
        }

        int total=0;
        cin >> k;
        for(int i=0; i<k; i++) {
            int a, b, c;
            cin >> a >> b >> c;

            int maxx = 0, maxv, side, counter;
            for(int j=0; j<G[a].size(); j++)
                if (G[a][j].c > maxx) {
                    maxx = G[a][j].c;
                    maxv = j;
                    side= a; counter = b;
```

```
                }

            for(int j=0; j<G[b].size(); j++)
                if (G[b][j].c > maxx) {
                    maxx = G[b][j].c;
                    maxv = j;
                    side= b; counter = a;
                }

            if (maxx <= c) continue;
            total = maxx-c;
            G[side][maxv].v = counter;
            G[side][maxv].c = c;
        }

        cin >> m;
        for(int i=0; i<m; i++) {
            int a, b, c;
            cin >> a >> b >> c;
        }

        if (t++) cout << endl;
        cout << before << endl << before-total << endl;
    }
    return 0;
}
```

graphs/1056.cpp

```
#include <iostream>
#include <cstring>
#include <string>
#include <map>
#include <cassert>
#define MAX 51

using namespace std;

int G[MAX][MAX];
int n, m;
map<string, int> M;

int person(string& s) {
    if (M.find(s) != M.end())
        return M[s];
    else
        return M[s]=M.size();

}

int main() {
    int t=0;
    while(cin >> n >> m, n|m) {
        memset(G, 0x1f, sizeof(G));
        M.clear();

        for(int i=0; i<m; i++) {
            string p, q;
            cin >> p >> q;
            int a = person(p), b=person(q);
            G[a][b] = G[b][a] = 1;
        }
        for(int i=1; i<=n; i++) G[i][i] = 0;

        assert(M.size() <= n);

        for(int k=1; k<=n; k++)
            for(int i=1; i<=n; i++)
```

```
                for(int j=1; j<=n; j++)
                    G[i][j] = min(G[i][j], G[i][k] + G[k][j]);

        int maxx = 0;
        for(int i=1; i<=n; i++)
                for(int j=1; j<=n; j++)
                    maxx = max(maxx, G[i][j]);

        cout << "Network " << ++t << ": ";

        if (maxx <= n)
            cout << maxx << endl;
        else
            cout << "DISCONNECTED" << endl;

        cout << endl;


    }
    return 0;
}
```

graphs/1174.cpp

```
#include <iostream>
#include <cstring>
#include <climits>
#include <string>
#include <vector>
#include <algorithm>
#include <queue>
#include <map>
#define MAX 200010

using namespace std;

struct Road {
    int v, c;
    Road(int v, int c) : v(v), c(c) {}
    inline bool operator < (const Road& that) const { return c > that.c; }
};

vector<Road> G[MAX];
priority_queue<Road> Q;
int n, m;
bool V[MAX];
map<string, int> M;

int city(string& s) {
    if (M.find(s) != M.end())
        return M[s];
    else
        return M[s]=M.size();

}

int main() {
    int t; cin >> t; t=0;

    while(cin >> n >> m) {
        memset(V, 0, sizeof(V));
        memset(G, 0, sizeof(G));
        M.clear();
        Q = priority_queue<Road>();

        for(int i=0; i<m; i++) {
            string p, q; int a, b, c;
            cin >> p >> q >> c;
```

```
            a = city(p); b=city(q);
            G[a].push_back(Road(b, c));
            G[b].push_back(Road(a, c));
        }

        int total = 0, totalc=0;

        Q.push(Road(1, 0));

        while(totalc < n) {
            Road item = Q.top(); Q.pop();
            if (V[item.v]) continue;

            V[item.v] = true;
            total += item.c;
            totalc++;

            for(int j=0; j<G[item.v].size(); j++)
                if (!V[G[item.v][j].v])
                    Q.push(G[item.v][j]);
        }

        if (t++) cout << endl;
        cout << total << endl;
    }
    return 0;
}
```

graphs/1197.cpp

```
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
using namespace std;

vector<int> G[501], P[30001];
bool VG[501], VP[30001];
int n, m;

int dfs(int v) {
    int sum = 1;
    VP[v] = true;

    for(int i=0; i<P[v].size(); i++) {
        int g = P[v][i];
        if (VG[g]) continue;
        VG[g] = true;

        for(int j=0; j<G[g].size(); j++) {
            int u = G[g][j];
            if (VP[u]) continue;
            sum += dfs(u);
        }
    }

    return sum;
}

int main() {
    while(cin >> n >> m, n|m) {
        memset(G, 0, sizeof(G));
        memset(P, 0, sizeof(P));
        memset(VG, 0, sizeof(VG));
        memset(VP, 0, sizeof(VP));

        for(int i=0; i<m; i++) {
            int k; cin >> k;
```

```
39              while(k--) {
40                  int a; cin >> a;
41                  G[i].push_back(a);
42                  P[a].push_back(i);
43              }
44          }
45
46          cout << dfs(0) << endl;
47      }
48      return 0;
49  }
```

graphs/1208.cpp

```
 1  #include <iostream>
 2  #include <cstring>
 3  #include <climits>
 4  #include <vector>
 5  #include <algorithm>
 6  #define MAX 501
 7
 8  using namespace std;
 9
10  int G[MAX][MAX], n;
11  bool V[MAX];
12  int D[MAX], DO[MAX];
13
14  struct Item {
15      int p, a, b;
16      Item(){}
17      Item(int p, int a, int b) : p(p), a(min(a,b)), b(max(a,b)) {}
18  };
19
20  bool comp(const Item& a, const Item& b) {
21      if (a.p != b.p) return a.p < b.p;
22      if (a.a != b.a) return a.a < b.a;
23      if (a.b != b.b) return a.b < b.b;
24      return false;
25  }
26
27  vector<Item> R;
28
29  int updateD(int i) {
30      D[i] = 0;
31      for(int j=0; j<n; j++) {
32          if (G[i][j] && G[i][j] < D[j]) {
33              D[j] = G[i][j];
34              DO[j] = i;
35          }
36      }
37  }
38
39  int main() {
40      int t; char skip;
41      cin >> t;
42      t = 0;
43      while(cin >> n) {
44          memset(V, 0, sizeof(V));
45          memset(D, 0x3F, sizeof(D));
46          R.clear();
47
48          for(int i=0; i<n; i++) {
49              for(int j=0; j<n; j++) {
50                  cin >> G[i][j];
51                  if (j+1<n) cin >> skip;
52              }
53          }
54
```

```
        int total = 0;

        V[0] = true;
        updateD(0);

        for(int k=1; k<n; k++) {
            int minn=INT_MAX, minv;
            for(int i=0; i<n; i++) {
                if (!V[i] && D[i] < minn) {
                    minn = D[i];
                    minv = i;
                }
            }
            R.push_back(Item(minn, DO[minv], minv));
            V[minv] = true;
            updateD(minv);
            total += minn;
        }

        sort(R.begin(), R.end(), comp);
        cout << "Case " << ++t << ":" << endl;
        for(int i=0; i<R.size(); i++) {
            cout << (char)(R[i].a+'A') << "-" << (char)(R[i].b+'A') << " " << R[i].p << endl;
        }
    }
    return 0;
}
```

graphs/1216.cpp

```
#include <iostream>
#include <cstring>
#include <cmath>
#define MAX 1000
using namespace std;

double G[MAX][MAX];
int X[MAX], Y[MAX], n, k;
int V[MAX];

void dfs(int v, int comp, int max) {
    V[v] = comp;
    for(int i=0; i<n; i++) {
        if (!V[i] && G[v][i] <= max)
            dfs(i, comp, max);
    }
}

int main() {
    int t; cin >> t; t=0;
    while(cin >> k) {
        n=0;

        double maxd=0;
        while(cin >> X[n], X[n]!=-1) {
            cin >> Y[n];
            for(int i=0; i<n; i++) {
                G[i][n] = G[n][i] = sqrt(pow(X[n]-X[i], 2.0)+pow(Y[n]-Y[i], 2.0));
                maxd = max(maxd, G[i][n]);
            }
            n++;
        }

        int begin=0, end=(int)ceil(maxd);
        int best, last = -1;
        while(begin <= end) {
            int mid = (begin+end)/2;
            if (mid == last) break;
```

```
39
40              int comp=0;
41              memset(V, 0, sizeof(V));
42              for(int i=0; i<n; i++)
43                  if (!V[i])
44                      dfs(i, ++comp, mid);
45
46              last = mid;
47              if (comp > k)
48                  begin = mid;
49              else {
50                  if (comp == k) best = mid;
51                  end = mid;
52              }
53          }
54
55          cout << best << endl;
56      }
57  }
```

graphs/1220.cpp

```
 1  #include <iostream>
 2  #include <string>
 3  #include <map>
 4  #include <cstring>
 5  #include <vector>
 6  #define MAX 205
 7  using namespace std;
 8
 9  map<string, int> E;
10  int emp(string& s) {
11      if (E.find(s) != E.end())
12          return E[s];
13      else
14          return E[s] = E.size()-1;
15  }
16
17  bool L[MAX], L2[MAX];
18  vector<int> G[MAX];
19  int n;
20
21  int dfs(int v) {
22      int acum = 0, illu = 0;
23      for(int i=0;i<G[v].size();i++) {
24          acum += dfs(G[v][i]);
25          if (L[G[v][i]]) illu++;
26      }
27      if (G[v].size() > 0 && illu < G[v].size())
28          L[v] = true;
29      return acum + L[v];
30  }
31
32  int main()
33  {
34      while(cin >> n, n) {
35          memset(G, 0, sizeof(G));
36          memset(L, 0, sizeof(L));
37          E.clear();
38          string a, b;
39          cin >> a;
40          emp(a);
41          for(int i=1;i<n;i++) {
42              cin >> a >> b;
43              G[emp(b)].push_back(emp(a));
44          }
45
46          int total = dfs(0);
```

```
47
48          memcpy(L2, L, sizeof(L));
49          bool unique = true;
50          for(int i=0; i<n; i++) {
51              if (!L2[i]) {
52                  memset(L, 0, sizeof(L));
53                  L[i] = true;
54                  if (dfs(0) == total) {
55                      unique = false;
56                      break;
57                  }
58              }
59          }
60
61          cout << n-total << " " << (unique?"Yes":"No") << endl;
62      }
63      return 0;
64  }
```

graphs/1229.cpp

```
 1  #include <iostream>
 2  #include <map>
 3  #include <string>
 4  #include <cstring>
 5  #include <sstream>
 6  #include <set>
 7  #include <algorithm>
 8  #define MAX 101
 9  using namespace std;
10
11  map<string, int> P;
12  int word(const string& p) {
13      if (P.find(p) != P.end())
14          return P[p];
15      else
16          return P[p] = P.size();
17  }
18
19  int O[MAX], npv, CO[MAX], GR[MAX];
20  string W[MAX];
21  bool G[MAX][MAX], V[MAX];
22  int n;
23  set<int> words;
24  set<string> answer;
25
26  void DFS(int v){
27      V[v] = true;
28      for(int i = 1; i <= n; i++)
29          if (G[v][i] && !V[i])
30              DFS(i);
31      O[npv++] = v;
32  }
33
34  int DFSt(int v, int comp){
35      int acum = 1;
36      V[v] = true; CO[v] = comp;
37      for(int i = 1; i <= n; i++)
38         if (G[i][v] && !V[i])
39               acum += DFSt(i, comp);
40      return acum;
41  }
42
43  void DFSf(int v){
44      V[v] = true;
45      answer.insert(W[v]);
46      for(int i = 1; i <= n; i++)
47          if (G[v][i] && !V[i])
```

```
48              DFSf(i);
49  }
50
51  int main() {
52      string s, p, q;
53      while(cin >> n, n) {
54          memset(G, 0, sizeof(G));
55          memset(CO, 0, sizeof(CO));
56          memset(GR, 0, sizeof(GR));
57          P.clear();
58          words.clear();
59          answer.clear();
60          getline(cin, p);
61
62          for(int i=0;i<n; i++) {
63              getline(cin, s);
64              stringstream sin(s);
65              sin >> p;
66              while(sin >> q) {
67                  G[word(p)][word(q)] = true;
68                  GR[word(p)]++;
69              }
70              W[word(p)] = p;
71          }
72
73          npv = 1;
74          memset(V, 0, sizeof(V));
75          memset(O, 0, sizeof(O));
76
77          for(int i = 1; i <= n; i++)
78              if(!V[i]) DFS(i);
79
80          memset(V, 0, sizeof(V));
81
82          int comp = 0;
83          for(int i = n; i > 0; i--) {
84              if(!V[O[i]]) {
85                  comp++;
86                  if (DFSt(O[i], comp) > 1 || GR[O[i]] == 0) {
87                      for(int j=1;j<=n;j++) {
88                          if (CO[j] == comp) {
89                              words.insert(j);
90                          }
91                      }
92                  }
93              }
94          }
95
96          memset(V, 0, sizeof(V));
97          for(set<int>::iterator it=words.begin(); it!=words.end(); it++) {
98              DFSf(*it);
99          }
100
101         cout << answer.size() << endl;
102         for(set<string>::iterator it=answer.begin(); it!=answer.end(); it++)
103             cout << (it!=answer.begin()?" ":"") << *it;
104         cout << endl;
105     }
106
107     return 0;
108 }
```

graphs/1234.cpp

```
1   #include <iostream>
2   #include <cstring>
3   #include <climits>
4   #include <vector>
```

```
 5  #include <algorithm>
 6  #include <queue>
 7  #define MAX 10005
 8
 9  using namespace std;
10
11  struct Road {
12      int v, c;
13      Road(int v, int c) : v(v), c(c) {}
14      inline bool operator < (const Road& that) const { return c < that.c; }
15  };
16
17  vector<Road> G[MAX];
18  int CStart[MAX], CCount[MAX], nc;
19  priority_queue<Road> Q;
20  vector<int> R;
21  int n, m;
22  bool V[MAX];
23
24  int dfs(int v) {
25      V[v] = true;
26      int acum = 1;
27      for(int i=0; i<G[v].size(); i++)
28          if (!V[G[v][i].v])
29              acum += dfs(G[v][i].v);
30      return acum;
31  }
32
33  int main() {
34      int t; cin >> t;
35      while(cin >> n >> m, t--) {
36          memset(V, 0, sizeof(V));
37          memset(G, 0, sizeof(G));
38          nc = 0;
39          R.clear();
40
41          for(int i=0; i<m; i++) {
42              int a, b, c;
43              cin >> a >> b >> c;
44              G[a].push_back(Road(b, c));
45              G[b].push_back(Road(a, c));
46          }
47
48          for(int i=1; i<=n; i++) {
49              if (!V[i]) {
50                  CStart[nc]=i;
51                  CCount[nc]=dfs(i);
52                  nc++;
53              }
54          }
55
56          int result=0;
57          for(int i=0; i<nc; i++) {
58              int totalc=0;
59              Q.push(Road(CStart[i], 0));
60              memset(V, 0, sizeof(V));
61
62              while(totalc < CCount[i]) {
63                  Road item = Q.top(); Q.pop();
64                  if (V[item.v]) { result+=item.c; continue; }
65
66                  V[item.v] = true;
67                  totalc++;
68
69                  for(int j=0; j<G[item.v].size(); j++)
70                      if (!V[G[item.v][j].v])
71                          Q.push(G[item.v][j]);
72              }
73              while(!Q.empty()) {
74                  result += Q.top().c;
```

```
                Q.pop();
            }
        }
        cout << result << endl;
    }
    return 0;
}
```

graphs/1235.cpp

```
#include <iostream>
#include <cstring>
#include <climits>
#define MAX 501

using namespace std;

int abs(int a) {
    return a>0?a:-a;
}

int d(int a, int b) {
    int result = 0;
    for(int i=0; i<4; i++) {
        int aa=a%10, bb=b%10;
        result += min(abs(aa-bb), 10-abs(aa-bb));
        a/=10; b/=10;
    }
    return result;
}

int K[MAX], G[MAX][MAX], n;
bool V[MAX];
int D[MAX];

int updateD(int i) {
    D[i] = 0;
    for(int j=0; j<n; j++) {
        if (G[i][j]) D[j] = min(D[j], G[i][j]);
    }
}

int main()
{
    int t;
    cin >> t;
    while(cin >> n) {
        memset(V, 0, sizeof(V));
        memset(D, 0x3F, sizeof(D));

        for(int i=0; i<n; i++)
            cin >> K[i];

        for(int i=0; i<n; i++)
            for(int j=i+1; j<n; j++)
                G[i][j] = G[j][i] = d(K[i],K[j]);

        int total=INT_MAX;
        for(int i=0;i<n;i++) total = min(total, d(0, K[i]));

        V[0] = true;
        updateD(0);

        for(int k=1; k<n; k++) {
            int minn=INT_MAX, minv;
            for(int i=0; i<n; i++) {
                if (!V[i] && D[i] < minn) {
                    minn = D[i];
```

```
                    minv = i;
                }
            }
            V[minv] = true;
            updateD(minv);
            total += minn;
        }

        cout << total << endl;
    }
    return 0;
}
```

graphs/1247.cpp

```
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#define MAX 30

using namespace std;

struct Edge {
    int u, v, c;
    Edge(int u, int v, int c) : u(u), v(v), c(c) {}
    inline bool operator < (const Edge& that) const { return c > that.c; }
};

int G[MAX][MAX];
int V[MAX];
int D[MAX];
int n, m;

void show(int t) {
    if (D[t] != t) {
        show(D[t]);
        cout << " ";
    }

    cout << (char)(t+'A');
}

int shortest(int a, int b) {
    memset(V, 0x3f, sizeof(V));
    priority_queue<Edge> Q;
    Q.push(Edge(a, a, 0));

    while(!Q.empty()) {
        Edge item = Q.top(); Q.pop();
        if (item.c >= V[item.v]) continue;
        V[item.v] = item.c;
        D[item.v] = item.u;

        for(int j=0; j<n; j++) {
            if (G[item.v][j]) {
                Edge e = Edge(item.v, j, item.c+G[item.v][j]);
                if (e.c <= V[e.v])
                    Q.push(e);
            }
        }
    }
    show(b); cout << endl;
}

int main() {
```

```
    while(cin >> n >> m) {
        memset(G, 0, sizeof(G));

        for(int i=0; i<m; i++) {
            char a, b; int c;
            cin >> a >> b >> c;
            G[a-'A'][b-'A'] = G[b-'A'][a-'A'] = c;
        }

        int k; cin >> k;
        while(k--) {
            char a, b; cin >> a >> b;
            shortest(a-'A', b-'A');
        }
    }
    return 0;
}
```

graphs/1263.cpp

```
#include <iostream>
#include <cstring>
#define MAX 2001
using namespace std;

int n;
bool V[MAX], G[MAX][MAX];
int X[MAX], Y[MAX], D[MAX], O[MAX], npv;

inline int abs(int a) {
    return a>0?a:-a;
}

void dfs(int v, bool sort){
    V[v] = true;
    for(int i = 1; i <= n; i++)
        if (G[v][i] && !V[i])
            dfs(i, sort);
    if (sort)
        O[++npv] = v;
}

int main() {
    int t; cin >> t; t=0;
    while(cin >> n) {
        memset(G, 0, sizeof(G));
        for(int i=1; i<=n; i++) {
            cin >> X[i] >> Y[i] >> D[i];
        }
        for(int i=1; i<=n; i++) {
            for(int j=1; j<=n; j++) {
                int r = D[i]/2;
                if (abs(X[j]-X[i])<=r && abs(Y[j]-Y[i]) <=r && i!=j)
                    G[i][j] = true;
            }
        }

        npv = 0;
        memset(V, 0, sizeof(V));
        memset(O, 0, sizeof(O));

        for(int i = 1; i <= n; i++)
            if(!V[i]) dfs(i, true);

        memset(V, 0, sizeof(V));

        int comp = 0;
        for(int i = n; i > 0; i--)
```

```
            if(!V[O[i]]) {
                comp++;
                dfs(O[i], false);
            }

        cout << comp << endl;
    }

    return 0;
}
```

graphs/10044.cpp

```
#include <iostream>
#include <cstring>
#include <climits>
#include <string>
#include <vector>
#include <queue>
#include <map>
#define MAX 5000
using namespace std;

vector<int> G[MAX];
int n, m;
bool V[MAX];
map<string, int> A;

struct Step {
    int x, v;
    Step() {}
    Step(int x, int v) : x(x), v(v) {}
};

queue<Step> Q;

int author(const string& a) {
    if (A.find(a) != A.end())
        return A[a];
    else
        return A[a] = A.size()-1;
}


char C[MAX];
void parseAuthors(const string& s) {
    vector<int> TA;
    int commas = 0, chars=0;
    for(int i=0;i<s.size();i++) {
        char c = s[i];
        if (chars == 0 && c == ' ') continue;

        if ((c==',' || c==':') && ++commas == 2) {
            TA.push_back(author(string(C, chars)));
            chars = commas = 0;
        } else {
            C[chars++] = c;
        }
    }
    for(int i=0;i<TA.size(); i++) {
        for(int j=i+1;j<TA.size(); j++) {
            G[TA[i]].push_back(TA[j]);
            G[TA[j]].push_back(TA[i]);
        }
    }

}

```

```
56  int main() {
57      string s;
58      int t=0, tt;
59      cin >> tt;
60      while(t++ < tt) {
61          cin >> n >> m;
62          memset(G, 0, sizeof(G));
63          A.clear();
64          getline(cin, s);
65          while(n--) {
66              getline(cin, s);
67              parseAuthors(s);
68          }
69
70          cout << "Scenario " << t << endl;
71          for(int i=0;i<m;i++) {
72              bool stop;
73              memset(V, 0, sizeof(V));
74              getline(cin, s);
75              int b = author(s);
76              Q = queue<Step>();
77              Q.push(Step(author("Erdos, P."), 0));
78              bool found = false;
79
80              while(!Q.empty()) {
81                  Step it = Q.front(); Q.pop();
82                  if (it.x == b) {
83                      cout << s << " " << it.v << endl;
84                      found = true;
85                      break;
86                  }
87
88                  V[it.x] = true;
89
90                  for(int i=0; i<G[it.x].size(); i++)
91                      if (!V[G[it.x][i]]) Q.push(Step(G[it.x][i], it.v+1));
92              }
93              if (!found) cout << s << " infinity" << endl;
94          }
95      }
96      return 0;
97  }
```

graphs/10092.cpp

```
1   #include <iostream>
2   #include <iomanip>
3   #include <cstring>
4   #include <string>
5   #include <cmath>
6   #include <climits>
7   #define MAX 1100
8   using namespace std;
9
10  int G[MAX][MAX], nk, np, n;
11  bool V[MAX];
12
13  int SOURCE() { return 1; }
14  int P(int i) { return 1+i; }
15  int K(int i) { return 1+np+i; }
16  int TARGET() { return 2+np+nk; }
17
18  int send(int s, int t, int minn) {
19      V[s] = true;
20
21      if (s==t) return minn;
22      for(int i=1; i<=n; i++) {
23          if (!V[i] && G[s][i] > 0) {
```

```
            if (int sent = send(i, t, min(minn, G[s][i]))) {
                G[s][i] -= sent;
                G[i][s] += sent;
                return sent;
            }
        }
    }
    return 0;
}

int main() {
    int tmp, tmp2;
    while(cin >> nk >> np, nk|np) {
        n = nk+np+2;
        int expected = 0;
        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));

        for(int i=1; i<=nk; i++) {
            cin >> tmp;
            expected += tmp;
            G[K(i)][TARGET()] = tmp;
        }

        for(int i=1;i<=np; i++) {
            cin >> tmp;
            G[SOURCE()][P(i)] = 1;
            for(int j=0;j<tmp;j++) {
                cin >> tmp2;
                G[P(i)][K(tmp2)] = 1;
            }
        }

        int total = 0;
        while(int sent = send(SOURCE(), TARGET(), INT_MAX)) {
            total += sent;
            memset(V, 0, sizeof(V));
        }
        cout << (expected == total ? 1: 0) << endl;
        if (expected == total) {
            for(int i=K(1); i<=K(nk); i++) {
                bool printed = false;
                for(int j=P(1); j<=P(np); j++) {
                    if (G[i][j]) {
                        cout << (printed?" ":"") << (j-1);
                        printed = true;
                    }
                }
                cout << endl;
            }
        }
    }
}
```

graphs/10243.cpp

```
#include <iostream>
#include <iomanip>
#include <cstring>
#include <queue>
#include <cmath>
#define MAX 1005
using namespace std;

int n;
bool G[MAX][MAX];
bool L[MAX];
bool V[MAX];
```

```
13
14  void dfs(int v, bool start) {
15      //cout << "*" << v << endl;
16      if (V[v]) return;
17      V[v] = true;
18      bool all = true;
19      int children = 0;
20      for(int i=0;i<n;i++) {
21          if (G[v][i] && !V[i]) {
22              dfs(i, false);
23              all &= L[i];
24              children++;
25          }
26      }
27      if (!all && children > 0 || start && children==0)
28          L[v] = true;
29  }
30
31  int main() {
32      int a, b, m;
33
34      while(cin >> n, n) {
35          memset(G, 0, sizeof(G));
36          memset(L, 0, sizeof(L));
37          memset(V, 0, sizeof(V));
38
39          for(int i=0;i<n;i++) {
40              cin >> m;
41              for(int j=0; j<m; j++) {
42                  cin >> a;
43                  a--;
44                  G[i][a] = G[a][i] = true;
45              }
46          }
47
48          int count = 0;
49          for(int i=0;i<n;i++) {
50              dfs(i, true);
51              if (L[i]) count++;
52          }
53
54          cout << count << endl;
55      }
56
57
58
59  }
```

graphs/10278.cpp

```
1   #include <iostream>
2   #include <iomanip>
3   #include <cstring>
4   #include <string>
5   #include <sstream>
6   #include <cmath>
7   #include <climits>
8   #define MAX 502
9   using namespace std;
10
11  int G[MAX][MAX], f, n;
12  bool F[MAX];
13
14  int main() {
15      int t; cin >> t;
16      string s;
17      while(t--) {
18          cin >> f >> n;
```

```
        memset(G, 0x3F, sizeof(G));
        memset(F, 0, sizeof(F));

        for(int i=0;i<f; i++) {
            int a; cin >> a; F[a] = true;
        }
        getline(cin, s);
        while(getline(cin, s), cin && s!="") {
            int a, b, c;
            stringstream inter(s);
            inter >> a >> b >> c;
            G[a][b] = G[b][a] = c;
        }


        for(int k=1; k<=n; k++) {
            G[k][k] = 0;
            for(int i=1; i<=n; i++)
                for(int j=1; j<=n; j++)
                    G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
        }

        int minn = INT_MAX, minv;
        for(int i=1; i<=n; i++) {
            int maxx = 0;
            for(int j=1; j<=n; j++) {
                int nearest = INT_MAX;
                for(int k=1; k<=n; k++) {
                    if (!F[k] && k!=i) continue;
                    nearest = min(nearest, G[k][j]);
                }
                maxx = max(maxx, nearest);
            }
            if (maxx < minn) {
                minn = maxx;
                minv = i;
            }
        }
        cout << minv << endl;
        if (t) cout << endl;
    }
}
```

graphs/10389.cpp

```
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#include <cmath>
#include <sstream>
#include <string>
#include <iomanip>
#include <cassert>
#define MAX 205
#define WALK 1
#define METRO 4

using namespace std;

struct Edge {
    int v; double c;
    Edge(int v, double c) : v(v), c(c) {}
    inline bool operator < (const Edge& that) const { return c > that.c; }
};

```

```
24  vector<Edge> G[MAX];
25  double V[MAX];
26  double X[MAX], Y[MAX];
27  int n;
28
29
30  double dist(double ax, double ay, double bx, double by) {
31      return sqrt(pow(ax-bx, 2.0) + pow(ay-by, 2.0))*60/10000;
32  }
33
34  int main() {
35      int t; cin >> t; t=0;
36      while(cin >> X[0] >> Y[0] >> X[1] >> Y[1]) {
37          memset(G, 0, sizeof(G));
38
39          G[0].push_back(Edge(1, dist(X[0], Y[0], X[1], Y[1])));
40          G[1].push_back(Edge(0, dist(X[0], Y[0], X[1], Y[1])));
41
42          n = 2;
43          string s;
44          getline(cin, s);
45          while(getline(cin, s) && s!="" && s[0]!=' ') {
46              stringstream sin(s);
47              int mn=0;
48              while(sin >> X[n] >> Y[n]) {
49                  if (X[n] == -1 && Y[n] == -1) {
50                      assert(mn >= 2);
51                      mn = 0;
52                      break;
53                  }
54                  if (mn > 0) {
55                      double mDist = dist(X[n-1], Y[n-1], X[n], Y[n])/METRO;
56                      G[n-1].push_back(Edge(n, mDist));
57                      G[n].push_back(Edge(n-1, mDist));
58                  }
59                  for(int i=0;i<n;i++) {
60                      double aDist = dist(X[n], Y[n], X[i], Y[i])/WALK;
61                      G[i].push_back(Edge(n, aDist));
62                      G[n].push_back(Edge(i, aDist));
63                  }
64
65                  n++; mn++;
66              }
67
68          }
69
70          int totalc=0;
71
72          for(int i=0; i<n; i++) V[i] = -1;
73
74          priority_queue<Edge> Q;
75          Q.push(Edge(0, 0));
76
77          while(totalc < n && !Q.empty()) {
78              Edge item = Q.top(); Q.pop();
79              if (item.c >= V[item.v] && V[item.v] >= 0) continue;
80
81              V[item.v] = item.c;
82              totalc++;
83
84              for(int j=0; j<G[item.v].size(); j++) {
85                  Edge e = G[item.v][j];
86                  if (item.c + e.c < V[e.v] || V[e.v] == -1)
87                      Q.push(Edge(e.v, item.c + e.c));
88              }
89          }
90
91          if (t++) cout << endl;
92          cout << (int)round(V[1]) << endl;
93      }
```

```
94      return 0;
95  }
```

graphs/10397.cpp

```
 1  #include <iostream>
 2  #include <cstring>
 3  #include <climits>
 4  #include <vector>
 5  #include <algorithm>
 6  #include <queue>
 7  #include <cmath>
 8  #include <iomanip>
 9  #define MAX 200010
10
11  using namespace std;
12
13  struct Road {
14      int v; double c;
15      Road(int v, double c) : v(v), c(c) {}
16      inline bool operator < (const Road& that) const { return c > that.c; }
17  };
18
19
20  vector<Road> G[MAX];
21  int X[MAX], Y[MAX];
22  priority_queue<Road> Q;
23  int n, m;
24  bool V[MAX];
25
26
27  int main() {
28      while(cin >> n) {
29          memset(V, 0, sizeof(V));
30          memset(G, 0, sizeof(G));
31          Q = priority_queue<Road>();
32
33          for(int i=1; i<=n; i++) {
34              int x, y;
35              cin >> x >> y;
36              X[i] = x; Y[i] = y;
37              for(int j=1; j<=i; j++) {
38                  double d = sqrt(pow(X[i]-X[j], 2.0)+pow(Y[i]-Y[j], 2.0));
39                  G[i].push_back(Road(j, d));
40                  G[j].push_back(Road(i, d));
41              }
42          }
43
44          cin >> m;
45          for(int i=0; i<m; i++) {
46              int a, b;
47              cin >> a >> b;
48              G[a].push_back(Road(b, 0));
49              G[b].push_back(Road(a, 0));
50          }
51
52          double total = 0; int totalc=0;
53          Q.push(Road(1,0));
54
55          while(totalc < n && !Q.empty()) {
56              Road item = Q.top(); Q.pop();
57              if (V[item.v]) continue;
58
59              V[item.v] = true;
60              total += item.c;
61              totalc++;
62
63              for(int j=0; j<G[item.v].size(); j++)
```

```
                if (!V[G[item.v][j].v])
                    Q.push(G[item.v][j]);
        }

        cout << setprecision(2);
        cout << fixed << total << endl;
    }
    return 0;
}
```

graphs/10480.cpp

```
#include <iostream>
#include <iomanip>
#include <cstring>
#include <string>
#include <cmath>
#include <climits>
#define MAX 1006
using namespace std;

int G[MAX][MAX], O[MAX][MAX], n, m;
bool V[MAX];

int send(int s, int t, int minn) {
    V[s] = true;

    if (s==t) return minn;
    for(int i=1; i<=n; i++) {
        if (!V[i] && G[s][i] > 0) {
            if (int sent = send(i, t, min(minn, G[s][i]))) {
                G[s][i] -= sent;
                G[i][s] += sent;
                return sent;
            }
        }
    }
    return 0;
}

int main() {
    int tt=0;
    while(cin >> n >> m, n|m) {
        if (tt++) cout << endl;

        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));
        memset(O, 0, sizeof(O));

        for(int i=0;i<m;i++) {
            int a, b, f;
            cin >> a >> b >> f;
            G[a][b] = G[b][a] += f;
            O[a][b] += f;
        }

        int total = 0;
        while(int sent = send(1, 2, INT_MAX)) {
            total += sent;
            memset(V, 0, sizeof(V));
        }
        for(int i=1;i<=n;i++) {
            for(int j=1;j<=n;j++) {
                if (O[i][j] > 0 && V[i] != V[j])
                    cout << i << " " << j << endl;
            }
        }
    }
```

```
}
```

graphs/10511.cpp

```
#include <iostream>
#include <iomanip>
#include <cstring>
#include <sstream>
#include <string>
#include <cmath>
#include <map>
#include <climits>
#define MAX 1300
using namespace std;

int G[MAX][MAX], n;
bool V[MAX];
map<string, int> EC, EP, EM;

int SOURCE() { return 1; }
int P(string& s) { if (EP.find(s)!=EP.end()) return EP[s]; else { return EP[s]=++n;} }
int M(string& s) { if (EM.find(s)!=EM.end()) return EM[s]; else { return EM[s]=++n;} }
int C(string& s) { if (EC.find(s)!=EC.end()) return EC[s]; else { return EC[s]=++n;} }
int TARGET() { return 2; }

int send(int s, int t, int minn) {
    V[s] = true;

    if (s==t) return minn;
    for(int i=1; i<=n; i++) {
        if (!V[i] && G[s][i] > 0) {
            if (int sent = send(i, t, min(minn, G[s][i]))) {
                G[s][i] -= sent;
                G[i][s] += sent;
                return sent;
            }
        }
    }
    return 0;
}

int main() {
    int t; cin >> t;
    string s, sm, sp, sc;
    getline(cin, s); getline(cin, s);
    while(t--) {
        EC.clear(); EP.clear(); EM.clear();
        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));
        n=2;

        while(getline(cin, s) && s!="" && s!=" ") {
            stringstream sin(s);
            sin >> sm >> sp;
            G[P(sp)][M(sm)] = 1;
            while(sin >> sc) {
                G[M(sm)][C(sc)] = 1;
                G[C(sc)][TARGET()] = 1;
            }
        }

        int maxParty = (EC.size()-1)/2;
        for(map<string, int>::iterator it=EP.begin(); it!=EP.end(); it++) {
            G[SOURCE()][it->second] = maxParty;
        }


        int total = 0;
```

```
        while(int sent = send(SOURCE(), TARGET(), INT_MAX)) {
            total += sent;
            memset(V, 0, sizeof(V));
        }

        if (total == EC.size()) {
            for(map<string, int>::iterator i=EM.begin(); i!=EM.end(); i++) {
                for(map<string, int>::iterator j=EC.begin(); j!=EC.end(); j++) {
                    if (G[j->second][i->second]) {
                        cout << i->first << " " << j->first << endl;
                    }
                }
            }
        } else {
            cout << "Impossible." << endl;
        }

        if (t) cout << endl;
    }
}
```

graphs/10986.cpp

```
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#define MAX 200010

using namespace std;

struct Edge {
    int v, c;
    Edge(int v, int c) : v(v), c(c) {}
    inline bool operator < (const Edge& that) const { return c > that.c; }
};

vector<Edge> G[MAX];
priority_queue<Edge> Q;
int n, m, s, t;
int V[MAX];


int main() {
    int tt; cin >> tt; tt=0;
    while(cin >> n >> m >> s >> t) {
        int before = 0;
        memset(V, 0x3f, sizeof(V));
        memset(G, 0, sizeof(G));
        Q = priority_queue<Edge>();

        for(int i=0; i<m; i++) {
            int a, b, c;
            cin >> a >> b >> c;
            G[a].push_back(Edge(b, c));
            G[b].push_back(Edge(a, c));
            before += c;
        }

        int totalc=0;

        Q.push(Edge(s, 0));

        while(totalc < n && !Q.empty()) {
            Edge item = Q.top(); Q.pop();
            if (item.c >= V[item.v]) continue;
```

```
46
47              V[item.v] = item.c;
48              totalc++;
49
50              for(int j=0; j<G[item.v].size(); j++) {
51                  Edge e = G[item.v][j];
52                  if (item.c + e.c < V[e.v])
53                      Q.push(Edge(e.v, item.c + e.c));
54              }
55          }
56
57          cout << "Case #" << ++tt << ": ";
58          if (V[t] < 0x3f3f3f3f)
59              cout << V[t] << endl;
60          else
61              cout << "unreachable" << endl;
62      }
63      return 0;
64  }
```

graphs/11110.cpp

```
1   #include <iostream>
2   #include <string>
3   #include <sstream>
4   #include <cstring>
5   #define MAX 102
6   using namespace std;
7
8   int G[MAX][MAX];
9   int n;
10
11  int fill(int x, int y, int v) {
12      if (G[x][y] != v) return 0;
13      if (x<=0 || x>n || y<=0 || y>n) return 0;
14
15      G[x][y] = -1;
16      return 1 +
17          fill(x-1, y, v) + fill(x+1, y, v) +
18          fill(x, y-1, v) + fill(x, y+1, v);
19  }
20
21  int main() {
22      while(cin >> n, n) {
23          int a, b; string s;
24          memset(G, 0, sizeof(G));
25
26          getline(cin, s);
27          for(int i=1;i<n;i++) {
28              getline(cin, s);
29              stringstream sin(s);
30              while(sin >> a >> b)
31                  G[a][b] = i;
32          }
33          bool good = true;
34          for(int i=1;i<=n;i++) {
35              for(int j=1;j<=n;j++) {
36                  if (G[i][j] >= 0)
37                      good &= fill(i,j, G[i][j]) == n;
38              }
39          }
40
41          cout << (good?"good":"wrong") << endl;
42
43      }
44      return 0;
45  }
```

graphs/11419.cpp

```cpp
#include <iostream>
#include <cstring>
#include <climits>
#include <string>
#include <cstdio>
#include <vector>
#define MAX 2005
using namespace std;

string VA[MAX], VB[MAX];
int G[MAX][MAX], n, r, c, p;
vector<int> G2[MAX];
bool V[MAX];

inline int SOURCE() { return 0; }
inline int TARGET() { return 1; }
inline int R(int i) { return 1+i; }
inline int C(int i) { return 1+r+i; }

int send(int s, int t, int minn) {
    V[s] = true;

    if (s==t) return minn;
    for(int i=0; i<G2[s].size(); i++) {
        int u = G2[s][i];
        if (!V[u] && G[s][u] > 0) {
            if (int sent = send(u, t, min(minn, G[s][u]))) {
                G[s][u] -= sent;
                G[u][s] += sent;
                return sent;
            }
        }
    }
    return 0;
}

void mark(int v, bool side) {
    V[v] = true;
    for(int i=0; i<G2[v].size(); i++) {
        int u = G2[v][i];
        if (!V[u] && (side && G[v][u] || !side && G[u][v]))
            mark(i, !side);
    }
}


int main() {
    while(scanf("%d %d %d", &r, &c, &p), r|c|p) {
        memset(G, 0, sizeof(G));
        memset(G2, 0, sizeof(G2));
        memset(V, 0, sizeof(V));

        for(int i=1; i<=r; i++) {
            G[SOURCE()][R(i)] = 1;
            G2[SOURCE()].push_back(R(i));
        }

        for(int i=1; i<=c; i++) {
            G[C(i)][TARGET()] = 1;
            G2[C(i)].push_back(TARGET());
        }


        for(int i=0; i<p; i++) {
            int a, b;
            cin >> a >> b;
```

```
67              G[R(a)][C(b)] = 1;
68              G2[R(a)].push_back(C(b));
69              G2[C(b)].push_back(R(a));
70          }
71
72          n = r+c+1;
73
74          int total = 0;
75          while(int sent = send(SOURCE(), TARGET(), INT_MAX)) {
76              total += sent;
77              memset(V, 0, sizeof(V));
78          }
79
80          V[0] = V[1] = true;
81          for(int i=1; i<=r; i++) {
82              bool inflow = false;
83              for(int j=1; j<=c; j++)
84                  inflow |= G[C(j)][R(i)];
85
86              if (!V[R(i)] && !inflow)
87                  mark(R(i), true);
88          }
89          printf("%d", total);
90          for(int i=1; i<=r; i++)
91              if (!V[R(i)]) printf(" r%d", i);
92
93          for(int i=1; i<=c; i++)
94              if (V[C(i)]) printf(" c%d", i);
95
96          printf("\n");
97      }
98
99      return 0;
100 }
```

graphs/11518.cpp

```
1  #include <iostream>
2  #include <vector>
3  #include <cstring>
4  #define MAX 10002
5  using namespace std;
6
7  vector<int> G[MAX];
8  bool V[MAX];
9  int n,m,l;
10
11 int dfs(int v) {
12     if (V[v]) return 0;
13     V[v] = true;
14     int r = 1;
15     for(int i=0;i<G[v].size(); i++)
16         r+=dfs(G[v][i]);
17     return r;
18 }
19
20 int main() {
21     int t; cin >> t;
22     while(cin >> n >> m >> l) {
23         memset(G, 0, sizeof(G));
24         memset(V, 0, sizeof(V));
25
26         for(int i=0;i<m;i++) {
27             int a, b;
28             cin >> a >> b;
29             G[a].push_back(b);
30         }
31         int sum = 0;
```

```
        for(int i=0;i<l;i++) {
            int a;
            cin >> a;
            sum+=dfs(a);
        }

        cout << sum << endl;

    }
    return 0;
}
```

graphs/11631.cpp

```
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#define MAX 200010

using namespace std;

struct Road {
    int v, c;
    Road(int v, int c) : v(v), c(c) {}
    inline bool operator < (const Road& that) const { return c > that.c; }
};

vector<Road> G[MAX];
priority_queue<Road> Q;
int n, m;
bool V[MAX];


int main() {
    while(cin >> n >> m, n|m) {
        int before = 0;
        memset(V, 0, sizeof(V));
        memset(G, 0, sizeof(G));
        Q = priority_queue<Road>();

        for(int i=0; i<m; i++) {
            int a, b, c;
            cin >> a >> b >> c;
            G[a].push_back(Road(b, c));
            G[b].push_back(Road(a, c));
            before += c;
        }

        int total = 0, totalc=0;

        Q.push(Road(0, 0));

        while(totalc < n) {
            Road item = Q.top(); Q.pop();
            if (V[item.v]) continue;

            V[item.v] = true;
            total += item.c;
            totalc++;

            for(int j=0; j<G[item.v].size(); j++)
                if (!V[G[item.v][j].v])
                    Q.push(G[item.v][j]);
        }

```

```
55          cout << before-total << endl;
56      }
57      return 0;
58  }
```

graphs/11686.cpp

```
 1  #include <iostream>
 2  #include <cstdio>
 3  #include <vector>
 4  #include <cstring>
 5  #define MAX 1000001
 6  using namespace std;
 7
 8  int V[MAX];
 9  int O[MAX], npv;
10  vector<int> G[MAX];
11  int n, m;
12
13  bool DFS(int d, int v){
14      V[v] = 1;
15
16      for(int i=0;i<G[v].size(); i++) {
17          int u = G[v][i];
18          if (V[u] == 1) return false;
19          if (!V[u] && !DFS(d, u)) return false;
20      }
21      O[++npv] = v;
22      V[v] = 2;
23      return true;
24  }
25
26
27  int main() {
28      int a, b;
29      while(scanf("%d%d",&n, &m), n|m) {
30          for(int i=1;i<=n;i++) G[i].clear();
31          npv = 0;
32          memset(V, 0, sizeof(V));
33          memset(O, 0, sizeof(O));
34
35          while(m--) {
36              scanf("%d%d",&a, &b);
37              G[a].push_back(b);
38          }
39
40
41          bool ok = true;
42          int d = 0;
43          for(int i = 1; i <= n; i++)
44              if (!V[i])
45                  ok &= DFS(++d, i);
46
47          if (ok)
48              for(int i = n; i > 0; i--)
49                  printf("%d\n", O[i]);
50          else
51              printf("IMPOSSIBLE\n");
52      }
53
54      return 0;
55  }
```

graphs/11709.cpp

```
1  #include <iostream>
```

```
 2  #include <map>
 3  #include <string>
 4  #include <cstring>
 5  #define MAX 1001
 6  using namespace std;
 7
 8  map<string, int> P;
 9  int person(const string& p) {
10      if (P.find(p) != P.end())
11          return P[p];
12      else
13          return P[p] = P.size();
14  }
15
16  bool V[MAX];
17  int O[MAX], npv;
18  bool G[MAX][MAX];
19  int n, m;
20
21  void DFS(int v){
22      V[v] = true;
23      for(int i = 1; i <= n; i++)
24          if (G[v][i] && !V[i])
25              DFS(i);
26      O[++npv] = v;
27  }
28
29  void DFSt(int v){
30      V[v] = true;
31      for(int i = 1; i <= n; i++)
32         if (G[i][v] && !V[i])
33              DFSt(i);
34  }
35
36
37  int main() {
38      int a, b, t; string p, q;
39      while(cin >> n >> m, n|m) {
40          memset(G, 0, sizeof(G));
41          P.clear();
42          getline(cin, p);
43
44          for(int i=0; i<n; i++) getline(cin, p);
45
46          while(m--) {
47              getline(cin, p);
48              getline(cin, q);
49              G[person(p)][person(q)] = true;
50          }
51
52          npv = 0;
53          memset(V, 0, sizeof(V));
54          memset(O, 0, sizeof(O));
55
56          for(int i = 1; i <= n; i++)
57              if(!V[i]) DFS(i);
58
59          memset(V, 0, sizeof(V));
60
61          int comp = 0;
62          for(int i = n; i > 0; i--)
63              if(!V[O[i]]) {
64                  comp++;
65                  DFSt(O[i]);
66              }
67
68          cout << comp << endl;
69      }
70
71      return 0;
```

```
}
```

graphs/11733.cpp

```
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#define MAX 10005

using namespace std;

struct Road {
    int v, c;
    Road(int v, int c) : v(v), c(c) {}
    inline bool operator < (const Road& that) const { return c > that.c; }
};

vector<Road> G[MAX];
int CStart[MAX], CCount[MAX], nc;
priority_queue<Road> Q;
int n, m, cca;
bool V[MAX];

int dfs(int v) {
    V[v] = true;
    int acum = 1;
    for(int i=0; i<G[v].size(); i++)
        if (!V[G[v][i].v])
            acum += dfs(G[v][i].v);
    return acum;
}

int main() {
    int t; cin >> t; t=0;
    while(cin >> n >> m >> cca) {
        memset(V, 0, sizeof(V));
        memset(G, 0, sizeof(G));
        nc = 0;

        for(int i=0; i<m; i++) {
            int a, b, c;
            cin >> a >> b >> c;
            if (c<cca) {
                G[a].push_back(Road(b, c));
                G[b].push_back(Road(a, c));
            }
        }

        for(int i=1; i<=n; i++) {
            if (!V[i]) {
                CStart[nc]=i;
                CCount[nc]=dfs(i);
                nc++;
            }
        }

        int total = nc*cca;

        for(int i=0; i<nc; i++) {
            int totalc = 0;
            Q = priority_queue<Road>();
            Q.push(Road(CStart[i], 0));
            memset(V, 0, sizeof(V));

            while(totalc < CCount[i]) {
```

```
                Road item = Q.top(); Q.pop();
                if (V[item.v]) continue;

                V[item.v] = true;
                total += item.c;
                totalc++;

                for(int j=0; j<G[item.v].size(); j++)
                    if (!V[G[item.v][j].v])
                        Q.push(G[item.v][j]);
            }
        }

        cout << "Case #" << ++t << ": " << total << " " << nc << endl;
    }
    return 0;
}
```

graphs/11747.cpp

```
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#define MAX 10005

using namespace std;

struct Road {
    int v, c;
    Road(int v, int c) : v(v), c(c) {}
    inline bool operator < (const Road& that) const { return c > that.c; }
};

vector<Road> G[MAX];
int CStart[MAX], CCount[MAX], nc;
priority_queue<Road> Q;
vector<int> R;
int n, m;
bool V[MAX];

int dfs(int v) {
    V[v] = true;
    int acum = 1;
    for(int i=0; i<G[v].size(); i++)
        if (!V[G[v][i].v])
            acum += dfs(G[v][i].v);
    return acum;
}

int main() {
    while(cin >> n >> m, n|m) {
        memset(V, 0, sizeof(V));
        memset(G, 0, sizeof(G));
        nc = 0;
        R.clear();

        for(int i=0; i<m; i++) {
            int a, b, c;
            cin >> a >> b >> c;
            G[a].push_back(Road(b, c));
            G[b].push_back(Road(a, c));
        }

        for(int i=1; i<=n; i++) {
            if (!V[i]) {
```

```
                CStart[nc]=i;
                CCount[nc]=dfs(i);
                nc++;
            }
        }

        for(int i=0; i<nc; i++) {
            int totalc=0;
            Q.push(Road(CStart[i], 0));
            memset(V, 0, sizeof(V));

            while(totalc < CCount[i]) {
                Road item = Q.top(); Q.pop();
                if (V[item.v]) { R.push_back(item.c); continue; }

                V[item.v] = true;
                totalc++;

                for(int j=0; j<G[item.v].size(); j++)
                    if (!V[G[item.v][j].v])
                        Q.push(G[item.v][j]);
            }
            while(!Q.empty()) {
                R.push_back(Q.top().c);
                Q.pop();
            }
        }
        sort(R.begin(), R.end());
        if (R.size()==0) {
            cout << "forest" << endl;
        } else {
            cout << R[0];
            for(int i=1; i<R.size(); i++)
                cout << " " << R[i];
            cout << endl;
        }
    }
    return 0;
}
```

graphs/11770.cpp

```
#include <iostream>
#include <vector>
#include <cstring>
#define MAX 10001
using namespace std;

bool V[MAX];
int O[MAX], npv;
vector<int> G[MAX];
int n, m;

void DFS(int v){
    if (V[v]) return;
    V[v] = true;
    for(int i = 0; i < G[v].size(); i++)
        DFS(G[v][i]);
    O[++npv] = v;
}

void DFSt(int v){
    if (V[v]) return;
    V[v] = true;
    for(int i = 0; i < G[v].size(); i++)
        DFSt(G[v][i]);
}

```

```
27
28  int main() {
29      int a, b;
30      int t; cin >> t; t=0;
31      while(cin >> n >> m) {
32          memset(G, 0, sizeof(G));
33
34          while(m--) {
35              cin >> a >> b;
36              G[a].push_back(b);
37          }
38
39          npv = 0;
40          memset(V, 0, sizeof(V));
41          memset(O, 0, sizeof(O));
42
43          for(int i = 1; i <= n; i++)
44              if(!V[i]) DFS(i);
45
46          memset(V, 0, sizeof(V));
47
48          int comp = 0;
49          for(int i = n; i > 0; i--)
50              if(!V[O[i]]) {
51                  comp++;
52                  DFSt(O[i]);
53              }
54
55          cout << "Case " << ++t << ": " <<  comp << endl;
56      }
57
58      return 0;
59  }
```

graphs/11833.cpp

```
1   #include <iostream>
2   #include <cstring>
3   #include <climits>
4   #include <vector>
5   #include <algorithm>
6   #include <queue>
7   #define MAX 252
8
9   using namespace std;
10
11  struct Edge {
12      int v, c;
13      Edge(int v, int c) : v(v), c(c) {}
14      inline bool operator < (const Edge& that) const { return c > that.c; }
15  };
16
17  int G[MAX][MAX];
18  int V[MAX], S[MAX];
19  int n, m, cc, kk;
20
21  int main() {
22      while(cin >> n >> m >> cc >> kk, n|m|cc|kk) {
23          memset(V, 0x3f, sizeof(V));
24          memset(S, 0, sizeof(S));
25          memset(G, -1, sizeof(G));
26
27          for(int i=0; i<m; i++) {
28              int a, b, c;
29              cin >> a >> b >> c;
30              G[a][b] = G[b][a] = c;
31          }
32
```

```
        for(int i=cc-2; i>=0; i--) {
            S[i] = S[i+1] + G[i][i+1];
        }

        int totalc=0;

        priority_queue<Edge> Q;
        Q.push(Edge(kk, 0));

        while(totalc < n && !Q.empty()) {
            Edge item = Q.top(); Q.pop();
            if (item.c >= V[item.v]) continue;
            V[item.v] = item.c;
            totalc++;
            if (item.v < cc) continue;
            for(int j=0; j<n; j++) {
                if (G[item.v][j]>=0) {
                    Edge e = Edge(j, G[item.v][j]);
                    if (item.c + e.c < V[e.v])
                        Q.push(Edge(e.v, item.c + e.c));
                }
            }
        }

        int minn = 0x3f3f3f3f;
        for(int i=0;i<cc;i++) {
            minn = min(minn, V[i]+S[i]);
        }
        cout << minn << endl;
    }
    return 0;
}
```

graphs/11838.cpp

```
#include <iostream>
#include <cstring>
#define MAX 1001
using namespace std;

bool V[MAX];
int O[MAX], npv;
bool G[MAX][MAX];
int n, m;

void DFS(int v){
    V[v] = true;
    for(int i = 1; i <= n; i++)
        if (G[v][i] && !V[i])
            DFS(i);
    O[++npv] = v;
}

void DFSt(int v){
    V[v] = true;
    for(int i = 1; i <= n; i++)
       if (G[i][v] && !V[i])
            DFSt(i);
}


int main() {
    int a, b, t;
    while(cin >> n >> m, n|m) {
        memset(G, 0, sizeof(G));

        while(m--) {
            cin >> a >> b >> t;
```

```
            G[a][b] = true;
            if (t==2)
                G[b][a] = true;
        }

        npv = 0;
        memset(V, 0, sizeof(V));
        memset(O, 0, sizeof(O));

        for(int i = 1; i <= n; i++)
            if(!V[i]) DFS(i);

        memset(V, 0, sizeof(V));

        int comp = 0;
        for(int i = n; i > 0; i--)
            if(!V[O[i]]) {
                comp++;
                DFSt(O[i]);
            }

        cout << (comp==1) << endl;
    }

    return 0;
}
```

graphs/12101.cpp

```
#include <iostream>
#include <queue>
#include <cstring>
#include <string>
using namespace std;

bool P[10000], V[10000];

struct Step {
    int a, b, c, d, w;
    Step() {}
    Step(int a, int b, int c, int d, int w) : a(a), b(b), c(c), d(d), w(w) {}

    int number() { return a*1000+b*100+c*10+d; }
    bool valid() { return a && P[number()]; }

    Step atA(int n) { return Step(n, b, c, d, w+1); }
    Step atB(int n) { return Step(a, n, c, d, w+1); }
    Step atC(int n) { return Step(a, b, n, d, w+1); }
    Step atD(int n) { return Step(a, b, c, n, w+1); }
};

Step makestep(int n) {
    int a, b, c, d;
    d = n%10; n/=10;
    c = n%10; n/=10;
    b = n%10; n/=10;
    a = n%10; n/=10;
    return Step(a,b,c,d,0);
}

int main() {
    memset(P, true, sizeof(P));
    P[0] = P[1] = false;
    for(int i=2; i<10000; i++) {
        if (P[i]) {
            for(int j=i*i; j<10000; j+=i)
                P[j] = false;
        }
```

```
    }
    int t, a, b;
    cin >> t;
    while(cin >> a >> b) {
        memset(V, 0, sizeof(V));
        queue<Step> Q;
        Q.push(makestep(a));
        bool found = false;
        while(!Q.empty()) {
            Step step = Q.front(); Q.pop();
            int n = step.number();
            if (V[n]) continue;
            V[n] = true;
            if (n == b) {
                cout << step.w << endl;
                found = true;
                break;
            }
            for(int i=0;i<=9;i++) {
                Step sa = step.atA(i);
                Step sb = step.atB(i);
                Step sc = step.atC(i);
                Step sd = step.atD(i);
                if (sa.valid()) Q.push(sa);
                if (sb.valid()) Q.push(sb);
                if (sc.valid()) Q.push(sc);
                if (sd.valid()) Q.push(sd);
            }
        }
        if (!found) cout << "Impossible" << endl;
    }

}
```

graphs/12135.cpp

```
#include <iostream>
#include <queue>
#include <cstring>
#include <string>
#define MAX 33000
using namespace std;

vector<int> G[MAX];
int V[MAX];

int n, m;
struct Step {
    int x, w;
    Step() {}
    Step(int x, int w) : x(x), w(w) {}
};

int main() {
    int t; cin >> t; int tt=0;
    while(cin >> n >> m, t--) {
        memset(G, 0, sizeof(G));
        memset(V, -1, sizeof(V));

        n = 1<<n;

        for(int i=0; i<m; i++) {
            int a, b, mask=0;
            cin >> a;
            while(a--) {
                cin >> b;
                mask = mask | (1<<b);
            }
```

```
            for(int i=0; i<n; i++)
                G[i].push_back(i^mask);
        }

        queue<Step> Q;
        Q.push(Step(0, 0));
        while(!Q.empty()) {
            Step step = Q.front(); Q.pop();
            if (V[step.x] >= 0) continue;

            V[step.x] = step.w;
            for(int i=0; i<G[step.x].size(); i++)
                Q.push(Step(G[step.x][i], step.w+1));
        }

        cout << "Case " << ++tt << ":" << endl;
        int q; string s;
        cin >> q;
        while(q--) {
            int b = 0;
            cin >> s;
            for(int i=0; i<s.size(); i++)
                b = b*2 + (s[i]-'0');

            cout << V[b] << endl;
        }
        cout << endl;
    }

}
```

graphs/12144.cpp

```
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#define MAX 501

using namespace std;

struct Edge {
    int u, v, c;
    Edge(int u, int v, int c) : u(u), v(v), c(c) {}
    inline bool operator < (const Edge& that) const { return c > that.c; }
};

int G[MAX][MAX];
int V[MAX];
vector<int> D[MAX];
int n, m, s, t;

void remove(int t) {
    if (D[t].size() == 0 || t == D[t][0]) return;
    for(int i=0; i<D[t].size(); i++) {
        G[D[t][i]][t] = 0;
        remove(D[t][i]);
    }
}

int shortest() {
    memset(V, 0x3f, sizeof(V));
    memset(D, 0, sizeof(D));
    priority_queue<Edge> Q;
    Q.push(Edge(s, s, 0));

```

```
    while(!Q.empty()) {
        Edge item = Q.top(); Q.pop();
        if (item.c > V[item.v]) continue;
        V[item.v] = item.c;
        D[item.v].push_back(item.u);

        for(int j=0; j<n; j++) {
            if (G[item.v][j]) {
                Edge e = Edge(item.v, j, item.c+G[item.v][j]);
                if (e.c <= V[e.v])
                    Q.push(e);
            }
        }
    }
    remove(t);
    if (V[t] < 0x3f3f3f3f)
        return V[t];
    else
        return -1;

}

int main() {
    while(cin >> n >> m, n|m) {
        cin >> s >> t;
        memset(G, 0, sizeof(G));

        for(int i=0; i<m; i++) {
            int a, b, c;
            cin >> a >> b >> c;
            G[a][b] = c;
        }

        shortest();
        cout << shortest() << endl;
    }
    return 0;
}
```

graphs/12159.cpp

```
#include <iostream>
#include <iomanip>
#include <cstring>
#include <vector>
#include <cmath>
#include <climits>
#include <vector>
#include <cassert>
#define MAX 306
using namespace std;

int X[MAX], Y[MAX], P[MAX], G[MAX][MAX], n, r, a, b;
bool V[MAX];

bool team(int c) {
    return (X[b] - X[a])*(Y[c] - Y[a]) - (Y[b] - Y[a])*(X[c] - X[a]) > 0;
}

int sqrdist(int a, int b) {
    return (X[a]-X[b])*(X[a]-X[b])+(Y[a]-Y[b])*(Y[a]-Y[b]);
}

int send(int s, int t, int minn) {
    V[s] = true;

    if (s==t) return minn;
    for(int i=0; i<=n; i++) {
```

```
28          if (!V[i] && G[s][i] > 0) {
29              if (int sent = send(i, t, min(minn, G[s][i]))) {
30                  G[s][i] -= sent;
31                  G[i][s] += sent;
32                  return sent;
33              }
34          }
35      }
36      return 0;
37  }
38
39  int main() {
40      int t=0;
41      while(cin >> n, n) {
42          memset(G, 0, sizeof(G));
43
44          for(int i=1;i<=n;i++)
45              cin >> X[i] >> Y[i] >> P[i];
46          cin >> a >> b >> r;
47
48          vector<int> A, B;
49          for(int i=1;i<=n;i++) {
50              if (P[i] == 0) continue;
51              if (team(i))
52                  B.push_back(i);
53              else
54                  A.push_back(i);
55          }
56          if (A.size() > B.size()) A.swap(B);
57
58          for(int i=0; i<A.size(); i++) {
59              int u=A[i];
60              G[0][u] = 1;
61              for(int j=0; j<B.size(); j++) {
62                  int v = B[j];
63                  G[v][n+1] = 1;
64                  if (sqrdist(u, v) <= r*r && P[u] > P[v])
65                      G[u][v] = 1;
66              }
67          }
68          n++;
69
70          memset(V, 0, sizeof(V));
71          int total = 0;
72          while(int sent = send(0, n, INT_MAX)) {
73              total += sent;
74              memset(V, 0, sizeof(V));
75          }
76          cout << "Case " << ++t << ": " << total << endl;
77      }
78  }
```

graphs/12160.cpp

```
1   #include <iostream>
2   #include <queue>
3   #include <cstring>
4   #include <string>
5   using namespace std;
6
7   bool V[10000];
8   int R[10];
9
10  struct Step {
11      int x, w;
12      Step() {}
13      Step(int x, int w) : x(x), w(w) {}
14
```

```
15      Step sum(int n) {
16          return Step((x+n)%10000, w+1);
17      }
18  };
19
20  int main() {
21      int a, b, n, t=0;
22      while(cin >> a >> b >> n, a|b|n) {
23          for(int i=0;i<n;i++)
24              cin >> R[i];
25
26          cout << "Case " << ++t << ": ";
27          memset(V, 0, sizeof(V));
28          queue<Step> Q;
29          Q.push(Step(a, 0));
30          bool found = false;
31          while(!Q.empty()) {
32              Step step = Q.front(); Q.pop();
33              if (V[step.x]) continue;
34              V[step.x] = true;
35              if (step.x == b) {
36                  cout << step.w << endl;
37                  found = true;
38                  break;
39              }
40              for(int i=0;i<n;i++)
41                  Q.push(step.sum(R[i]));
42          }
43          if (!found) cout << "Permanently Locked" << endl;
44      }
45
46  }
```

graphs/12168.cpp

```
 1  #include <iostream>
 2  #include <cstring>
 3  #include <climits>
 4  #include <string>
 5  #define MAX 505
 6  using namespace std;
 7
 8  string V1[MAX], V2[MAX];
 9  int G[MAX][MAX], n;
10  bool V[MAX];
11
12  int send(int s, int t, int minn) {
13      V[s] = true;
14
15      if (s==t) return minn;
16      for(int i=0; i<=n; i++) {
17          if (!V[i] && G[s][i] > 0) {
18              if (int sent = send(i, t, min(minn, G[s][i]))) {
19                  G[s][i] -= sent;
20                  G[i][s] += sent;
21                  return sent;
22              }
23          }
24      }
25      return 0;
26  }
27
28  int main() {
29      int t; cin >> t;
30      int c, d, v;
31
32      while(cin >> c >> d >> v, t--) {
33          memset(G, 0, sizeof(G));
```

```
34          memset(V, 0, sizeof(V));
35
36          string s1, s2;
37          for(int i=1; i<=v; i++) {
38              cin >> s1 >> s2;
39              V1[i] = s1; V2[i] = s2;
40
41              bool dog = s1[0] == 'D';
42
43              if (dog)
44                  G[0][i] = 1;
45              else
46                  G[i][v+1] = 1;
47
48              for(int j=1; j<i; j++) {
49                  if (s1 == V2[j] || s2 == V1[j])
50                      if (dog)
51                          G[i][j] = 1;
52                      else
53                          G[j][i] = 1;
54              }
55          }
56          n = v+1;
57
58          int total = 0;
59          while(int sent = send(0, n, INT_MAX)) {
60              total += sent;
61              memset(V, 0, sizeof(V));
62          }
63          cout << v-total << endl;
64      }
65
66      return 0;
67  }
```

graphs/12179.cpp

```
 1  #include <iostream>
 2  #include <cstring>
 3  #include <cmath>
 4  #include <iomanip>
 5  #define MAX 101
 6  using namespace std;
 7
 8  int G[MAX][MAX], n, r, c;
 9  double P[101][10001];
10
11  int main() {
12      int t; cin >> t; t=0;
13      cout << fixed << setprecision(6);
14
15      while(cin >> n >> r) {
16          memset(G, 0x3F, sizeof(G));
17          memset(P, 0, sizeof(P));
18
19          char cc;
20          for(int i=0; i<n; i++) {
21              for(int j=0; j<n; j++) {
22                  cin >> cc;
23                  if (cc=='Y') G[i][j] = 1;
24              }
25          }
26
27          for(int k=0; k<n; k++)
28              for(int i=0; i<n; i++)
29                  for(int j=0; j<n; j++)
30                      G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
31
```

```
        P[0][0] = 1;
        double pp = 1.0/r;
        for(int i=1; i<=100; i++)
            for(int k=1; k<=r; k++)
                for(int j=k; j<=100*r; j++)
                    P[i][j] += P[i-1][j-k] * pp;

        cout << "Case " << ++t << endl;
        cin >> c;
        while(c--) {
            int a, b, m;
            cin >> a >> b >> m;
            a--; b--;

            int d=G[a][b];

            double total = 0;
            for(int i=0; i<=m; i++)
                total += P[d][i];
            cout << total << endl;
        }
        cout << endl;
    }
    return 0;
}
```

graphs/12186.cpp

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstring>
#include <cmath>
#define MAX 100002
using namespace std;

vector<int> G[MAX];
int n, t;

int dfs(int v) {
    if (G[v].empty()) return 1;
    vector<int> mins;
    for(int i=0; i<G[v].size(); i++)
        mins.push_back(dfs(G[v][i]));
    sort(mins.begin(), mins.end());

    int get = (int)ceil(G[v].size()*t/100.0);
    int sum = 0;
    for(int i=0; i<get; i++) sum+=mins[i];
    return sum;
}

int main() {
    int boss;
    while(cin >> n >> t, n|t) {
        memset(G, 0, sizeof(G));
        for(int i=1; i<=n; i++) {
            cin >> boss; G[boss].push_back(i);
        }
        cout << dfs(0) << endl;
    }
    return 0;
}
```

Math

- GCD
  - UVa 12184 - Transcribed Books
- Sieve
  - UVa 1246 - Find Terrorists
- Prime Factorization
  - UVa 12137 - Puzzles of Triangles
- Geometry
  - UVa 12300 - Smallest Regular Polygon
  - UVa 12194 - Isosceles Triangles

math/1246.cpp

```cpp
#include <iostream>
#include <vector>
#include <cstring>
using namespace std;

bool P[100];
int T[10000001];
vector<int> W;

long long real_mod(long long a, long long b) {
    long long c = a%b;
    if (c<0) c+=b;
    return c;
}

int main() {
    int n, k;

    memset(P, true, sizeof(P));
    P[0] = P[1] = false;
    for(int i=2; i<100; i++) {
        if (P[i]) {
            W.push_back(i);
            for(int j=i*i; j>=0 && j<100; j+=i)
                P[j] = false;
        }
    }

    int t; cin >> t; t=0;
    int a, b;
    while(cin >> a >> b) {
        memset(T, 0, sizeof(int)*(b-a+1));

        if (a==0) { T[0]-=2; T[1] -= 1; }
        if (a==1) { T[0]-=1; }

        for(long long i=2; i*i<=b; i++) {
            for(long long j=max(real_mod(i*i+i-a, i), i*i+i-a); j<=(b-a); j+=i) {
                T[j]+=2;
            }
            int tmp = i*i-a;
            if (tmp >= 0 && tmp <= (b-a))
                T[tmp]++;
        }

        int cnt=0;
        for(int i=0; i<=(b-a);i++) {
            if (P[T[i]+2]) {
                cout << (cnt++?" ":"") << i+a;
            }
        }
        if (!cnt) cout << -1;
        cout << endl;
    }

}
```

math/12137.cpp

```cpp
#include <string.h>
#include <stdio.h>
#define ull unsigned long long

int W[] = { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83
```

```
 7  inline ull div(const ull& a, const ull& b, ull &r) {
 8      r = a/b;
 9      return a-r*b;
10  }
11
12  inline ull pow(const ull& a, const int b) {
13      if (b==0) return 1;
14      ull tmp = b&1 ? a : 1;
15      ull r = pow(a, b>>1);
16      return tmp*r*r;
17  }
18
19  int main() {
20      ull n;
21      int t=0;
22      while(scanf("%llu", &n), n) {
23          ull ncopy = n;
24          ull step = 1;
25          for(int i=0; ncopy>1 && i<wn; i++) {
26              int power=0;
27              ull divr;
28              while(div(ncopy, W[i], divr)==0) {
29                  ncopy = divr;
30                  power++;
31              }
32              step *= pow(W[i], (power+1)/2);
33          }
34          step *= ncopy;
35
36          ull result;
37          if (div(n, step, result)==0) result--;
38          result *= 8;
39
40          if(result)
41              printf("Case %d: %llu\n",++t, result);
42          else
43              printf("Case %d: Impossible\n", ++t);
44      }
45  }
```

math/12184.cpp

```
 1  #include <iostream>
 2  using namespace std;
 3
 4  long gcd(long a, long b) {
 5      while(b) {
 6          long c = a%b;
 7          a = b;
 8          b = c;
 9      }
10      return a;
11  }
12
13  int main() {
14      int t; cin >> t;
15      int n;
16      while(cin >> n) {
17          long result = 0;
18          long maxSerial = 0;
19          for(int i=0; i<n; i++) {
20              long s=0, tmp;
21              for(int j=0; j<9; j++) {
22                  cin >> tmp; s+=tmp;
23              }
24              cin >> tmp;
25              s -= tmp;
26              maxSerial = max(maxSerial, tmp);
```

```
            result = gcd(result, s);
        }
        if (result>1 && maxSerial < result)
            cout << result << endl;
        else
            cout << "impossible" << endl;
    }
}
```

math/12194.cpp

```
#include <cstdio>
#include <algorithm>
#include <cstring>
#define MAX 1010
using namespace std;

int X[MAX], Y[MAX];
long T[MAX][MAX];
int C[MAX];

inline long sqr(long v) { return v*v; }

int main(){
    int n;
    while(scanf("%d", &n), n) {
        memset(C, 0, sizeof(C));

        for(int i=0; i<n; i++)
            scanf("%d %d", &X[i], &Y[i]);

        int sum = 0;
        for(int i=0; i<n; i++) {
            for(int j=0; j<n; j++)
                T[i][C[i]++] = sqr(X[i]-X[j])+sqr(Y[i]-Y[j]);
            sort(T[i], T[i]+C[i]);
            long last=-1L;
            int cnt=0;
            for(int j=0; j<C[i]; j++) {
                if (T[i][j] != last) {
                    sum += cnt*(cnt-1)/2;
                    cnt = 0;
                }
                last = T[i][j];
                cnt++;
            }
            sum += cnt*(cnt-1)/2;
        }

        printf("%d\n", sum);
    }

}
```

math/12300.cpp

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define PI 3.14159265358979323846
using namespace std;

double cot(double angle) {
    return cos(angle)/sin(angle);
}

```

```
11  int main(){
12      int x1, y1, x2, y2, n;
13      while(cin >> x1 >> y1 >> x2 >> y2 >> n, x1 | y1 | x2 | y2 | n) {
14          double d = sqrt(pow(x2-x1, 2.0)+pow(y2-y1, 2.0));
15          int k = n/2;
16          double s = sin(PI/n)/sin(PI*k/n)*d;
17          double A = 0.25*n*s*s*cot(PI/n);
18          setprecision(6);
19          cout << fixed << A << endl;
20      }
21
22  }
```

Misc

- STL map
  - UVa 10420 - List of Conquests
  - UVa 11629 - Ballot evaluation
- String parsing
  - UVa 1200 - A DP Problem
- Priority queue
  - UVa 1203 - Argus
- Binary Manipulation
  - UVa 11532 - Simple Adjacency Maximization
- Binary Search
  - UVa 12190 - Electric Bill
  - UVa 12192 - Grapevine
  - UVa 1215 - String Cutting
- Segment Tree
  - UVa 1232 - SKYLINE
- Sort
  - UVa 11157 - Dynamic Frog
  - UVa 12189 - Dinner Hall
- Greed
  - UVa 12172 - Matchsticks
- Linked List
  - UVa 245 - Uncompress
- Union-Find
  - UVa 11503 - Virtual Friends
  - UVa 10158 - War
  - UVa 11966 - Galactic Bonding
- Ad hoc
  - UVa 11494 - Queen
  - UVa 11597 - Spanning Subtree
  - UVa 12148 - Electricity
  - UVa 12195 - Jingle Composing
  - UVa 12155 - ASCII Diamondi
  - UVa 12196 - Klingon Levels

misc/245.cpp

```cpp
#include <iostream>
#include <sstream>
#include <string>
#include <climits>
#include <cstring>
#include <cstdio>
#include <list>
#define MAX 1000
using namespace std;

list<string> W;
stringstream sin;
int curnum=0;
bool word=false, number=false;

void finishWord() {
    W.push_back(sin.str());
    sin.str("");
    word = false;
}

void finishNumber() {
    list<string>::iterator it = W.end();
    while(curnum--)
        it--;

    cout << *it;
    W.push_back(*it);
    W.erase(it);

    curnum = 0;
    number = false;
}

int main() {
    string s;

    while(getline(cin, s), s!="0") {
        for(int i=0; i<s.size(); i++) {
            char c = s[i];
            if (c >= 'a' && c <= 'z' || c >= 'A' && c <= 'Z') {
                sin << c;
                word = true;
            } else if (word) finishWord();

            if (c >= '0' && c <= '9') {
                curnum *= 10; curnum += c-'0';
                number = true;
            } else if (number) finishNumber();

            if (!number)
                cout << c;
        }
        if (word) finishWord();
        if (number) finishNumber();

        cout << endl;
    }
}
```

misc/1200.cpp

```cpp
#include <iostream>
#include <string>
#include <cmath>
```

```
4  using namespace std;
5
6  int getSign(string& s, int &i) {
7      if (s[i] == '+') { i++; return 1; }
8      if (s[i] == '-') { i++; return -1; }
9      return 1;
10 }
11
12 int getNumber(string& s, int& i, bool& got) {
13     int result = 0;
14     while(s[i] >='0' && s[i] <= '9') {
15         result = result*10 + (s[i]-'0');
16         i++;
17         got = true;
18     }
19     return result;
20 }
21
22 bool getX(string& s, int& i) {
23     return i<s.size() && s[i] == 'x' && ++i;
24 }
25
26 bool willChange(string& s, int& i) {
27     return i<s.size() && s[i] == '=' && ++i;
28 }
29
30
31 int main()
32 {
33     int t;
34     string s;
35     cin >> t;
36
37     while(cin >> s) {
38         int i=0, A=0, B=0, masterSign = 1;
39         while(i<s.size()) {
40             int sign = getSign(s, i);
41             bool got = false;
42             int number = getNumber(s, i, got);
43             bool isX = getX(s, i);
44             if (isX && !got) number = 1;
45            // cout << masterSign << " " << sign << " " << number << " " << isX << endl;
46
47             if (isX)
48                 B += -1*masterSign*sign*number;
49             else
50                 A += masterSign*sign*number;
51             if (willChange(s, i)) masterSign *= -1;
52         }
53         if (A==0 && B==0) {
54             cout << "IDENTITY" << endl;
55         } else if (B==0) {
56             cout << "IMPOSSIBLE" << endl;
57         } else {
58             cout << (int)floor(((double)A/B))<< endl;
59         }
60
61     }
62
63     return 0;
64 }
```

misc/1203.cpp

```
1  #include<cstdio>
2  #include<iostream>
3  #include<queue>
4  #include<string>
```

```cpp
using namespace std;

#define SZ 3200

struct Item{
  int p, q, b;

  Item() {}
  Item(int q, int p) : p(p), q(q), b(p) {}
  Item(int q, int p, int b) : p(p), q(q), b(b) {}

  inline bool operator < (const Item &d) const{
    if(this->p==d.p) return d.q<this->q;
    return this->p>d.p;
  }

  Item next() {
    return Item(q, p+b, b);
  }

};

priority_queue<Item> Q;
int q, p;

int main(void) {
  string s;
  int q, p, k;

  while(cin >> s, s!="#") {
    cin >> q >> p;
    Q.push(Item(q, p));
  }

  cin >> k;
  for(int i=0; i<k; ++i) {
    Item item = Q.top(); Q.pop();
    cout << item.q << endl;
    Q.push(item.next());
  }

  return 0;
}
```

misc/1215.cpp

```cpp
#include <iostream>
#include <string>
#include <set>
using namespace std;

int T[10001][26];
int C[1001];
set<int> K;

int main() {
    int t; cin >> t; t=0;
    int k; string s;
    while(cin >> k) {
        K.clear();
        for(int i=0; i<k; i++)
            cin >> C[i];

        cin >> s;
        for(int i=1; i<=s.size(); i++)
            for(int j=0; j<26; j++)
                T[i][j] = T[i-1][j] + (s[i-1] == j+'a');

```

```
23
24          K.insert(0);
25          K.insert(s.size());
26
27          int total = 0;
28          for(int i=0; i<k; i++) {
29              int mid = C[i];
30              set<int>::iterator it = K.lower_bound(mid);
31              int hi = *it; it--;
32              int lo = *it;
33
34              for(int j=0; j<26; j++) {
35                  int sidea = T[mid][j]-T[lo][j];
36                  int sideb = T[hi][j]-T[mid][j];
37
38                  if (sidea>0 ^ sideb>0) total++;
39              }
40              K.insert(mid);
41          }
42          cout << total << endl;
43      }
44  }
```

misc/1232.cpp

```
 1  #include <iostream>
 2  #include <string>
 3  #include <set>
 4  using namespace std;
 5
 6  struct Node {
 7      int a, b, h;
 8      bool leaf;
 9      Node() {}
10      Node(int a, int b, int h, bool leaf=true) : a(a), b(b), h(h), leaf(leaf) {}
11  };
12
13  Node H[5000005];
14  inline int left(int i) { return 2*i; }
15  inline int right(int i) { return 2*i+1; }
16
17  inline void cut(int v, int x) {
18      H[left(v)] = Node(H[v].a, x, H[v].h);
19      H[right(v)] = Node(x, H[v].b, H[v].h);
20      H[v].leaf = false;
21  }
22
23  int dfs(int v, int a, int b, int h) {
24      a = max(a, H[v].a);
25      b = min(b, H[v].b);
26      if (b<=a) return 0;
27
28      if (!H[v].leaf)
29          return dfs(left(v), a, b, h) + dfs(right(v), a, b, h);
30
31      if (H[v].h > h) return 0;
32      if (H[v].a < a) return cut(v, a), dfs(v, a, b, h);
33      if (b < H[v].b) return cut(v, b), dfs(v, a, b, h);
34
35      H[v].h = h;
36      return b-a;
37  }
38
39  int main() {
40      int n, t; cin >> t; t=0;
41      while(cin >> n, n) {
42          H[1] = Node(0, 100000, 0);
43
```

```
        int sum = 0;
        while(n--) {
            int a, b, h;
            cin >> a >> b >> h;
            sum += dfs(1, a, b, h);
        }
        cout << sum << endl;

    }
}
```

misc/10158.cpp

```
#include <iostream>
#include <map>
#include <string>
#include <cstring>
#include <algorithm>
using namespace std;

int P[20000];

inline int enemy(int v) { return v+10000; }

inline int findset(int v) {
    if (P[v] != -1 && P[v] != v)
        return P[v] = findset(P[v]);
    return v;
}

inline int unionset(int x, int y) {
    int a = findset(x), b = findset(y);
    if (a<b) swap(a,b);
    P[b] = a;
}

int main() {
    memset(P, -1, sizeof(P));
    int n, c, x, y;
    cin >> n;
    while(cin >> c >> x >> y, c|x|y) {
        if (c==1) {
            if (findset(x) == findset(enemy(y))) { cout << -1 << endl; continue; }
            unionset(x, y);
            unionset(enemy(x), enemy(y));
        } else if (c==2) {
            if (findset(x) == findset(y)) { cout << -1 << endl; continue; }
            unionset(x, enemy(y));
            unionset(enemy(x), y);
        } else if (c==3) {
            cout << (findset(x) == findset(y)) << endl;
        } else if (c==4) {
            cout << (findset(x) == findset(enemy(y))) << endl;
        }
    }
}
```

misc/10420.cpp

```
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#include <map>
#define MAX 105
using namespace std;
```

```
 8
 9  map<string, int> women;
10  int main() {
11      int n;
12      string s;
13      cin >> n;
14      while(n--) {
15          cin >> s;
16          women[s]++;
17          getline(cin, s);
18      }
19
20      for(map<string, int>::const_iterator it = women.begin(); it != women.end(); it++) {
21          cout << it->first << " " << it->second << endl;
22      }
23
24      return 0;
25  }
```

misc/11157.cpp

```
 1  #include <iostream>
 2  #include <algorithm>
 3  #include <vector>
 4  using namespace std;
 5
 6  vector<int> V;
 7
 8  int main() {
 9      int t, n, d;
10      cin >> t; t=0;
11
12      while(cin >> n >> d) {
13          char a; int b;
14          V.clear();
15          V.push_back(0);
16          V.push_back(d);
17          for(int i=0;i<n; i++) {
18              cin >> a; cin.ignore(); cin >> b;
19              V.push_back(b);
20              if (a=='B')
21                  V.push_back(b);
22          }
23          sort(V.begin(), V.end());
24
25          int maxx = 0;
26          for(int i=3;i<V.size(); i+=2)
27              maxx = max(maxx, V[i]-V[i-2]);
28
29          for(int i=2;i<V.size(); i+=2)
30              maxx = max(maxx, V[i]-V[i-2]);
31
32          cout << "Case " << ++t << ": " << maxx << endl;
33      }
34
35  }
```

misc/11494.cpp

```
 1  #include <iostream>
 2  #include <cstring>
 3  #include <iomanip>
 4  using namespace std;
 5
 6  int main() {
 7      int x, y, a, b;
```

```
    while(cin >> x >> y >> a >> b, x|y|a|b) {
        if (x==a && y==b)
            cout << 0 << endl;
        else if (x==a || y==b || x+y == a+b || x-y==a-b)
            cout << 1 << endl;
        else
            cout << 2 << endl;
    }
    return 0;
}
```

misc/11503.cpp

```
#include <iostream>
#include <map>
#include <string>
#include <cstring>
#include <algorithm>
using namespace std;

int P[200001], C[200001];
map<string, int> M;

int parent(int v) {
    if (P[v] != v) {
        int p = P[v] = parent(P[v]);
        C[v] = C[p];
        return p;
    } else {
        return v;
    }
}

int person(string& s) {
    if (M.find(s) != M.end())
        return M[s];
    else {
        int r = M[s] = M.size();
        C[r] = 1; P[r] = r;
        return r;
    }
}

int main() {
    int t; cin >> t; t=0;
    int n;

    while(cin >> n) {
        M.clear();
        while(n--) {
            string p, q;
            cin >> p >> q;
            int a = person(p), b=person(q);
            int pa = parent(a), pb=parent(b);
            if (pa==pb) {
                cout << C[pa] << endl;
                continue;
            }
            if (pa < pb) swap(pa, pb);

            P[pb] = pa;
            cout << (C[pa]+=C[pb]) << endl;
        }

    }
}
```

misc/11532.cpp

```cpp
#include <iostream>
#include <cstring>
#include <iomanip>
using namespace std;

long long T[51][51];

int main() {
    for(int i=1;i<=50;i++) {
        for(int j=0;i+j<=50;j++) {
            int p=i, q=j;
            long long n = 0L;
            if (p%2!=0 && p/2<q) {
                n = 1;
                p--;
            }

            for(;p>1;p-=2) {
                if (q>0)
                    n = (n<<3) | 5L;
                else
                    n = (n<<2) | 3L;
                q--;
            }

            if (p==1) n = (n<<1) | 1L;
            T[i][j] = n;
        }
    }


    int t, p, q;
    cin >> t;
    while(cin >> p >> q) {
        cout << T[p][q] << endl;
    }

    return 0;
}
```

misc/11597.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    int n, t=0;
    while(cin >> n, t++, n) {
        cout << "Case " << t << ": " << n/2 << endl;
    }

    return 0;
}
```

misc/11629.cpp

```cpp
#include <iostream>
#include <map>
#include <cstring>
using namespace std;

map<string, int> P;

```

```
 8  int main() {
 9      int n, g;
10      while(cin >> n >> g) {
11          P.clear();
12          for(int i=0; i<n; i++) {
13              string s; int a, b;
14              cin >> s >> a; cin.get(); cin >> b;
15              P[s] = a*10+b;
16          }
17
18          for(int i=1; i<=g; i++) {
19              string s = "+"; int d=0; int r;
20
21              while(s=="+") {
22                  cin >> s;
23                  d += P[s];
24                  cin >> s;
25              }
26              cin >> r; r*=10;
27
28              bool result;
29              if ( s=="<") result = d < r;
30              if ( s=="<=") result = d <= r;
31              if ( s==">") result = d > r;
32              if ( s==">=") result = d >= r;
33              if ( s=="=") result = d == r;
34              cout << "Guess #" << i << " was " << (result?"correct":"incorrect") << "." << endl;
35          }
36
37      }
38  }
```

misc/11966.cpp

```
 1  #include <iostream>
 2  #include <map>
 3  #include <string>
 4  #include <cstring>
 5  #include <algorithm>
 6  #include <cmath>
 7  using namespace std;
 8
 9  int P[1000];
10  double X[1000], Y[1000];
11
12  inline int findset(int v) {
13      if (P[v] == v) return v;
14      return P[v] = findset(P[v]);
15  }
16
17  inline bool unionset(int x, int y) {
18      int a = findset(x), b = findset(y);
19      if (a==b) return false;
20      P[b] = a;
21      return true;
22  }
23
24  inline double dist(int a, int b) {
25      return pow(X[a]-X[b], 2.0)+pow(Y[a]-Y[b], 2.0);
26  }
27
28  int main() {
29      int t; cin >> t; t=0;
30
31      int n; double d;
32      while(cin >> n >> d) {
33          for(int i=0; i<n; i++) P[i] = i;
34
```

```
        int sets = n;
        for(int i=0; i<n; i++) {
            cin >> X[i] >> Y[i];
            for(int j=0;j<i;j++)
                if (dist(i,j)<=d*d && unionset(i, j))
                    sets--;
        }

        cout << "Case " << ++t << ": " << sets << endl;
    }
}
```

misc/12148.cpp

```
#include <iostream>
using namespace std;

int M[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

bool oneday(int ad, int am, int ay, int bd, int bm, int by) {
    if (--bd == 0) {
        if (--bm == 0) {
            --by;
            bm=12;
        }

        bd = M[bm-1];

        bool isleap = (by%4==0 && (by%100!=0 || by%400==0));
        if (bm==2 && isleap) bd=29;
    }
    return ad==bd && am==bm && ay==by;

}

int main() {
    int n, ad=0, am=0, ay=0, ac=0;
    while(cin >> n, n) {
        int sum = 0, count=0;
        while(n--) {
            int bd, bm, by, bc;
            cin >> bd >> bm >> by >> bc;
            if (oneday(ad, am, ay, bd, bm, by)) {
                sum += bc-ac; count++;
            }
            ad = bd; am = bm; ay = by; ac = bc;
        }
        cout << count << " " << sum << endl;
    }

    return 0;
}
```

misc/12155.cpp

```
#include <iostream>
using namespace std;

inline int abs(int n) { return n>0?n:-n; }

inline char charAt(int n, int x, int y) {
    x%=n*2-1; y%=n*2-1;
    int dist = abs(n-x-1)+abs(n-y-1);
    if (dist < n)
        return (char)(dist%26+'a');
    else
```

```
12          return '.';
13      }
14
15      int main() {
16          int n, ax, ay, bx, by, t=0;
17          while(cin >> n, n) {
18              cin >> ax >> ay >> bx >> by;
19              cout << "Case " << ++t << ":" << endl;
20              for(int i=ax; i<=bx; i++) {
21                  for(int j=ay; j<=by; j++) {
22                      cout << charAt(n, i, j);
23                  }
24                  cout << endl;
25              }
26          }
27
28      }
```

misc/12172.cpp

```
1   #include <iostream>
2   #include <cstring>
3   #include <cmath>
4   #define MAX 101
5   using namespace std;
6
7   void printMax(int n) {
8       if (n&1) { cout << "7"; n-=3; }
9       for(;n;n-=2) cout << "1";
10  }
11
12  void printMin(int n) {
13      switch(n) {
14          case 2: cout << "1"; return;
15          case 3: cout << "7"; return;
16          case 4: cout << "4"; return;
17          case 5: cout << "2"; return;
18          case 6: cout << "6"; return;
19      }
20
21      switch(n%7) {
22          case 1: cout << "10"; n-=8; break;
23          case 2: cout << "1"; n-=2; break;
24          case 3:
25              if (n==10) {
26                  cout << "22"; n-= 10;
27              } else{
28                  cout << "200"; n-=17;
29              }
30              break;
31          case 4: cout << "20"; n-= 11; break;
32          case 5: cout << "2"; n-= 5; break;
33          case 6: cout << "6"; n-= 6; break;
34      }
35      for(;n;n-=7) cout << "8";
36  }
37
38  int main() {
39
40
41      int n;
42      int t; cin >> t; t=0;
43
44      while(cin >> n) {
45          printMin(n);
46          cout << " ";
47          printMax(n);
48          cout << endl;
```

```
49      }
50
51      return 0;
52  }
```

misc/12189.cpp

```
1   #include <iostream>
2   #include <vector>
3   #include <algorithm>
4   using namespace std;
5
6   struct Event {
7       int s; char t;
8       Event() {}
9       Event(int s, char t) : s(s), t(t) {}
10      int entry() { return t=='E'?1:0; }
11      int exit() { return t=='X'?1:0; }
12      int unknown() { return t=='?'?1:0; }
13  };
14
15  bool compare(const Event& a, const Event& b) {
16      return a.s < b.s;
17  }
18
19  vector<Event> V;
20
21  int main() {
22      int n;
23      while(cin >> n, n) {
24          int entries=0, exits=0, unknowns=0;
25          int a, b, c; char t;
26          V.clear();
27          for(int i=0; i<n; i++) {
28              cin >> a >> t >> b >> t >> c >> t;
29              Event e = Event(a*60*60+b*60+c, t);
30              entries += e.entry();
31              exits += e.exit();
32              unknowns += e.unknown();
33              V.push_back(e);
34          }
35          sort(V.begin(), V.end(), compare);
36
37          int maxEntries = (unknowns-(entries-exits))/2;
38          int maxx = 0, current=0;
39          for(int i=0; i<V.size(); i++) {
40              if (V[i].entry()) current++;
41              if (V[i].exit()) current--;
42              if (V[i].unknown()) {
43                  if (maxEntries) { current++; maxEntries--; }
44                  else { current--; }
45              }
46              maxx = max(maxx, current);
47          }
48          cout << maxx << endl;
49      }
50  }
```

misc/12190.cpp

```
1   #include <iostream>
2   using namespace std;
3
4   int C(int price) {
5       int cons = 0;
6       cons += min(max(0, price/2), 100); price -= 2*100;
```

```
    cons += min(max(0, price/3), 9900); price -= 3*9900;
    cons += min(max(0, price/5), 990000); price -= 5*990000;
    cons += max(0, price/7);
    return cons;
}

int V(int cons) {
    int price = 0;
    price += min(max(0, cons*2), 2*100); cons -= 100;
    price += min(max(0, cons*3), 3*9900); cons -= 9900;
    price += min(max(0, cons*5), 5*990000); cons -= 990000;
    price += max(0, cons*7);
    return price;
}

int main() {
    int a, b;
    while(cin >> a >> b,a|b) {
        int total = C(a);
        int begin = 0, end = total;
        int answer = 0;
        while(begin < end) {
            int mine = (begin+end)/2;
            int diff = V(total-mine)-V(mine);
            if (diff > b)
                begin = mine;
            else if (diff < b)
                end = mine;
            else { answer = mine; break; }
        }

        cout << V(answer) << endl;
    }

    return 0;
}
```

misc/12192.cpp

```
#include <iostream>
#include <cstring>
#include <vector>
#include <algorithm>
using namespace std;

int T[1001][501];
int S[1001];

int main() {
    int n, m, q;
    while(cin >> n >> m, n|m) {
        memset(S, 0, (m+n)*sizeof(int));

        for(int i=0; i<n; i++)
            for(int j=0; j<m; j++)
                cin >> T[i-j+m][S[i-j+m]++];

        cin >> q;
        while(q--) {
            int L, U;
            cin >> L >> U;
            int maxx = 0;
            for(int i=0;i<m+n; i++) {
                int a = lower_bound(T[i], T[i]+S[i], L) - T[i];
                int b = upper_bound(T[i], T[i]+S[i], U) - T[i];
                maxx = max(maxx, b-a);
            }
            cout << maxx << endl;
```

```
        }

        cout << "-" << endl;
    }
}
```

misc/12195.cpp

```
#include <iostream>
#include <string>
using namespace std;

int duration(char c) {
    switch(c) {
        case 'W': return 64;
        case 'H': return 32;
        case 'Q': return 16;
        case 'E': return 8;
        case 'S': return 4;
        case 'T': return 2;
        case 'X': return 1;
    }
}

int main() {
    string s;
    while(cin >> s, s!="*") {
        int d=0, r=0;
        for(int i=1; i<s.size(); i++) {
            if (s[i] == '/') {
                if (d==64) r++;
                d = 0;
                continue;
            }
            d+=duration(s[i]);
        }
        cout << r << endl;
    }

    return 0;
}
```

misc/12196.cpp

```
#include <iostream>
#include <climits>
#include <cstring>
using namespace std;

int T[10001][1001];
int N[10001];

inline long abs(long n) { return n>0?n:-n;}

int main() {
    int n, tmp;
    while(cin >> n, n) {
        memset(T, 0, n*sizeof(T[0]));
        for(int i=0; i<n; i++) {
            cin >> N[i];
            for(int j=0; j<N[i]; j++) {
                cin >> tmp;
                T[i][tmp]++;
            }
            for(int j=1;j<=1000;j++)
                T[i][j] += T[i][j-1];
```

```
23          }
24
25          long minn = INT_MAX;
26          for(int t=0;t<=1000;t++) {
27              long sum=0;
28              for(int i=0; i<n; i++) {
29                  sum += abs(N[i] - 2*T[i][t]);
30              }
31              minn = min(minn, sum);
32          }
33          cout << minn << endl;
34      }
35  }
```