Final exam 1:00~3:05 pm

Before you start the test, take 1~2 minutes to finish the following surveys (**mandatory**)

https://docs.google.com/forms/d/e/1FAIpQLSdDTlhLSIJElWzSUu1psmVG_xDRPDpIGTWXJdiHGh5ZtdDx1A/viewform?usp=sf_link (also in classroom)

For the Final:

https://xjoi.net/contest/4905

For your **3 best-performed problems**, you get up to 100 points each. The other 2 problems count as bonus, and you get up to **10 points each**. Don't spend too much time on bonus problems if you still have incomplete regular problems.

You have 2 hours to finish. Please work on them **independently**.

You can consult internet for C++ syntax, library functions or your previous homework. You can ask for clarifications for problem description. You can't ask questions for debugging issues or algorithms.

Final exam review

- Final exam results:
  - 9 students scored 300 or above; 6 scored 200~300 and 5 got 200 or below
- Please finish at least 60% of homework, midterm and final exams before Sunday 09/04 if you haven’t done so

8-Connected (P1227)

- Basic floodfill problem
- Need to consider 8 directions

```
3  char s[101][101];
4
5  void dfs(int x,int y)
6  {
7   s[x][y]='.'; //make it visited
8   for(int i=-1;i<=1;i++) // go through all neighbors
9     for(int j=-1;j<=1;j++) {
10       int dx=x+i,dy=y+j;
11       if(s[dx][dy]=='W')
12         dfs(dx,dy);
13      }
14     return ;
15 }
```

```
17 int main()
18 {
19   int n,m;
20   scanf("%d%d",&n,&m);
21
22     for(int i=0;i<n;i++)
23       for(int j=0;j<m;j++)
24         cin>>s[i][j];
25     int sum=0;
26     for(int i=0;i<n;i++)
27       for(int j=0;j<m;j++) {
28         if(s[i][j]=='W') {
29           sum++;
30           dfs(i,j);
31          }
32        }
33     cout<<sum<<endl;
34
35  return 0;
36 }
```

Relatives (P9324)

- Basic DSU problem
- Use methods we learned in last class

```
4  // basic DSU problem
5  int parent[5001];
6  int get(int x) {
7    if (x == parent[x])
8      return x;
9    return parent[x] = get(parent[x]);
10 }
11
12 bool same(int x, int y) {return get(x) == get(y);}
13
14 void unite(int a, int b){
15   a = get(a), b = get(b);
16   if (a == b) return;
17   // make sure point to smaller parent
18   if (parent[a] > parent[b]) swap(a, b);
19   parent[b] = a;
20 }
21
```

```
23 int main() {
24   int n, m, p;
25   cin >> n >> m >> p;
26   for(int i = 1; i <= n; i++)
27       parent[i] = i;
28
29   for (int i = 0; i < m; i++){
30     int a, b;
31     cin >> a >> b;
32     unite(a, b);
33   }
34
35   for (int i = 0; i < p; i++){
36     int a, b;
37     cin >> a >> b;
38     if (same(a, b)){
39       cout << "Yes" << endl;
40       continue;
41     }
42     cout << "No" << endl;
43   }
44 }
```

Roads of Caringness (P8019)

- Have to make a turn at every cow
- Keep and check previous direction
- Use DFS, need to end up at (0, 0)
- Backtracking when return

```cpp
#include<bits/stdc++.h>
#define maxn 10001
using namespace std;
int n,ans;
bool vis[maxn];
struct node{
    int x,y;
}a[maxn];

int main(){
    vector<int> vec;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
        scanf("%d%d",&a[i].x,&a[i].y);
    dfs(0,0,0,0);
    cout<<ans;
}
```

```cpp
//pre: 1-up, 2-down, 3-left, 4-right
void dfs(int x,int y,int cnt,int pre){
    if(cnt==n){ // make a turn and go back to origin
        if(x==0&&y>0&&pre!=2){ans++; return;}
        if(x==0&&y<0&&pre!=1){ans++; return;}
        if(y==0&&x>0&&pre!=3){ans++; return;}
        if(y==0&&x<0&&pre!=4){ans++; return;}
    }
    for(int i=1;i<=n;i++){ // check 4 directions
        if(vis[i])continue;
        // next cow is at right, and previous dir
        // is not right. This is a valid visit
        if(a[i].x>x&&a[i].y==y&&pre!=4){//go right
            vis[i]=1;
            dfs(a[i].x,a[i].y,cnt+1,4);
            vis[i]=0; //back tracking
        }
        if(a[i].x<x&&a[i].y==y&&pre!=3){//go left
            vis[i]=1;
            dfs(a[i].x,a[i].y,cnt+1,3);
            vis[i]=0; //back tracking
        }
        if(a[i].x==x&&a[i].y<y&&pre!=2){//go down
            vis[i]=1;
            dfs(a[i].x,a[i].y,cnt+1,2);
            vis[i]=0; //back tracking
        }
        if(a[i].x==x&&a[i].y>y&&pre!=1){//go up
            vis[i]=1;
            dfs(a[i].x,a[i].y,cnt+1,1);
            vis[i]=0; //back tracking
        }
    }
}
```

Chores (P1556)

- All dependencies are before current task
- Min cost of current task is max cost of dependencies plus cost of current task
- Compare previous overall min cost and min cost of current task

```cpp
#include<bits/stdc++.h>
using namespace std;

int n,l,t,ans[10005],maxans;
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;++i){
        scanf("%d%d",&i,&l);
        int tmp=0;
        // check max cost of dependencies
        while(scanf("%d",&t)&&t)
            tmp=max(ans[t],tmp);

        // max dependency plus my cost
        ans[i]=tmp+l;

        // overall cost
        maxans=max(ans[i],maxans);
    }
    printf("%d\n",maxans);
    return 0;
}
```

Chess game (P7906)

- Very similar to HW in Week 8
- Use a string to represent state
- BFS to search from start state to end state
- Check availability before swapping

```
4  typedef string state;
5  map<state, bool> visited;
6  int dx[4] = {-1, 1, -4, 4};
7
8  struct node {
9    state s;
10   int steps;
11 };
12
13 int main() {
14   node start, end, next;
15   int temp;
16   string tempstring;
17   // read starting and end nodes
18   for (int i = 0; i < 4; i++) {
19     cin >> tempstring;
20     start.s = start.s + tempstring;
21   }
22   for (int i = 0; i < 4; i++) {
23     cin >> tempstring;
24     end.s = end.s + tempstring;
25   }
26   start.steps = 0;
27
```

```
28   queue<node> q;
29   q.push(start);
30   visited[start.s] = true;
31
32   while (!q.empty()) {
33     if (q.front().s == end.s) {
34       cout << q.front().steps;
35       return 0;
36     }
37
38     for (int j = 0; j < 16; j++) {
39       for (int i = 0; i < 4; i++) {
40         // check available
41         if (i == 0 && j%4 == 0) continue;
42         if (i == 1 && j%4 == 3) continue;
43         if (i == 2 && j < 4) continue;
44         if (i == 3 && j > 11) continue;
45         next.s = q.front().s;
46         temp = next.s[j]; // smap two positions
47         next.s[j] = next.s[j+dx[i]];
48         next.s[j+dx[i]] = temp;
49         if (visited.count(next.s)!=0) continue;
50
51         visited[next.s]=true;
52         next.steps = q.front().steps + 1;
53         q.push(next);
54       }
55     }
56     q.pop();
57   }
```

HW11 - Friendship Circle (P9561)

- Use concept of DSU
- Define get() and unite() functions

```
4  int parent[201],arr[201][201],n,ans=0;
5
6  // regular get
7  int search(int x) {
8    if (x==parent[x])
9      return x;
10   return search(parent[x]);
11 }
12
13 // regular unite
14 bool merge(int x,int y){
15   x=search(x),y=search(y);
16   if (x==y)
17     return false;
18   parent[x]=y;
19   return true;
20 }
21
```

```
22 int main() {
23   cin >> n;
24   for (int i=0;i<n;i++)
25     parent[i]=i;
26   for (int i=0; i<n;i++)
27     for (int j=0;j<n;j++)
28     cin >> arr[i][j];
29
30   for (int i=0;i<n;i++)
31     for (int j=0;j<n;j++)
32       if (arr[i][j]==1) merge(i,j);
33
34   for (int i=0;i<n;i++) {
35     if (search(i)!=i) continue;
36     ans++;
37   }
38   cout << ans;
39 }
```

HW11 - DSU (P8214)

- DSU exercise using improvements

```
5  int set[10001] = { 0 }; // parent
6  int s[10001] = { 0 }; // rank
7
8  int get(int x){
9    if (set[x] == 0) return x;
10   return set[x] = get(set[x]);
11 }
12
13 // unite by rank
14 void unite(int a, int b){
15   int x = get(a);
16   int y = get(b);
17   if (x != y){
18     if (s[x] == s[y]) {
19       set[x] = y;
20       s[y]++;
21     }
22     else if (s[x] < s[y])
23       set[x] = y;
24     else
25       set[y] = x;
26   }
27 }
28
```

```
29 int main() {
30   int n, m;
31   cin >> n >> m;
32   int zi, xi, yi;
33   char ans[m];
34   int ansit = 0;
35   for (int i = 0; i < m; i++){
36     cin >> zi >> xi >> yi;
37     if (zi == 2){
38       if (get(xi) == get(yi)) ans[ansit] = 'Y';
39       else ans[ansit] = 'N';
40       ansit++;
41     }
42     else if (zi == 1){
43       unite(xi, yi);
44     }
45   }
46   for (int i = 0; i < ansit; i++){
47     printf("%c \n", ans[i]);
48   }
49 }
```

HW11 - Fence (P8483)

- DSU problem
- Construct rect while merging

```
5  struct rect {
6      int x1, y1, x2, y2; // two corners
7      int pl;
8  } fences[100001];
9
10 int p[100001]; // parent
11 int r[100001]; // rank
12 int ans = INT_MAX;
13
14 // find parent
15 int find_set(int v) {
16     return v == p[v] ? v : p[v] = find_set(p[v]);
17 }
18
19 // merge two rect and put in rect b (b is parent)
20 void merge_fences(int a, int b){
21     fences[b].x1 = min(fences[a].x1, fences[b].x1);
22     fences[b].y1 = min(fences[a].y1, fences[b].y1);
23     fences[b].x2 = max(fences[a].x2, fences[b].x2);
24     fences[b].y2 = max(fences[a].y2, fences[b].y2);
25 }
```

```
27 // unite then merge rect
28 void union_sets(int a, int b) {
29     a = find_set(a);
30     b = find_set(b);
31     if (a==b) return;
32
33     // choose parent by comparing rank
34     // after unite, b is always parent
35     if(r[a] == r[b]){
36         p[a] = b;
37         r[b]++;
38     }
39     else if (r[a] < r[b]){
40         p[a] = b;
41     }
42     else {
43         p[b] = a;
44         swap(a, b); // ensure b is parent
45     }
46     merge_fences(a, b);
47 }
48
```

```
49 int main(){
50     int N, M;
51     cin >> N >> M;
52     for(int i=1; i<=N; i++){
53         cin >> fences[i].x1 >> fences[i].y1;
54         fences[i].x2 = fences[i].x1; // make a dot rect
55         fences[i].y2 = fences[i].y1;
56         p[i] = i;
57         r[i] = 0;
58     }
59     for(int i=1; i<=M; i++){
60         int cowA, cowB;
61         cin >> cowA >> cowB;
62         union_sets(cowA, cowB);
63     }
64     for(int i=1; i<=N; i++){
65         if (i == find_set(i)){
66             ans = min(ans, 2*((fences[i].x2-fences[i].x1) +
67               (fences[i].y2-fences[i].y1)));
68         }
69     }
70     cout<<ans<<endl;
71 }
```