X-Camp

2022 Summer
Post-Midterm Parent Meeting

Class: 202B

Teacher: Nick Wang

TAs: Peter Li, Cory Fan

Midterm Test Summary

- Total 5 problems. Among them, 3 are regular problems (100 points each), and 2 are bonus (10 points each)
- 300 is considered full score; 320 is possible highest score
- 20 students completed midterm
  - 12 students got 300 or above; 5 got 200~300 and 3 got 150~200
- We are going to review all problems today. For students who didn't complete 100%, please finish them after today's class.
- Parents survey: https://docs.google.com/forms/d/e/1FAIpQLScen9oEQOkSP2f469jMnt5IGIwqhovY_6IBrvXYqZ5yH-LfUg/viewform?usp=sf_link

Homework Policy and TA Office Hours

- **Homework Policy:** Students are expected to complete homework **individually** and submit before the start of the next class. Required problems are mandatory and bonus problems are optional.
- **Important**: If can't finish in first week, students should finish them after review.

- Students are encouraged to have discussion with buddies
- **TA Office Hour** Students should come to TA Office Hour if they have questions for homework.

  Peter: Wednesday 7~8 PM

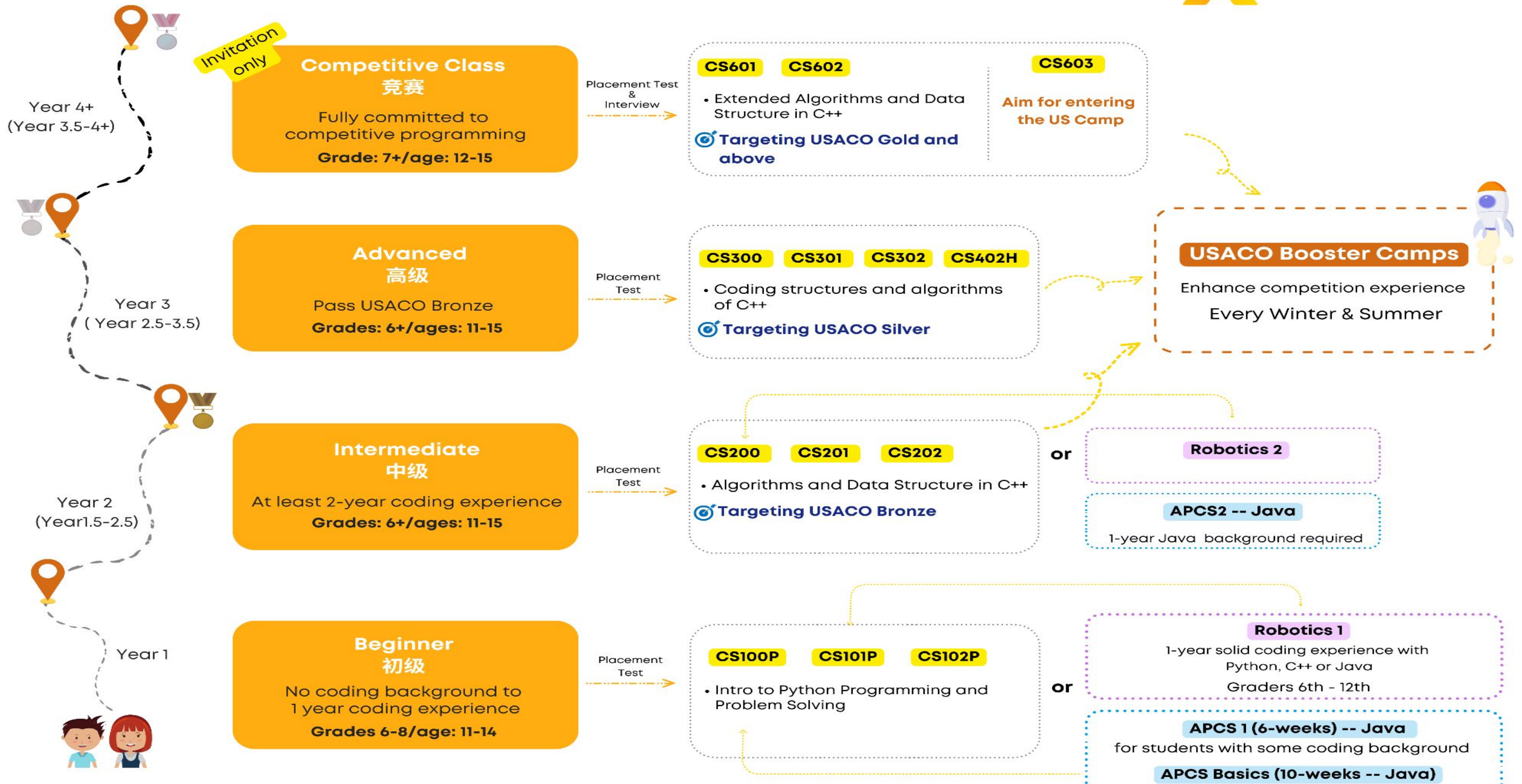
  Cory: Friday 7~8 PM

New Release of the 4.0 Award

- Students who get full scores on all homeworks and exams (required problems and bonus problems) are qualified for X-Camp 4.0 Award.
- All qualified students will be asked to upload a 2-min learning experience video. In this video, the students will be given a list of questions and asked to share their learning experience at X-Camp. We will send out an email notification to qualified students one week after the final exam with detailed instructions.
- Parents will be asked to sign a media consent waiver before submitting the video.
- Students will get an X-Camp E-Certificate as rewards.
- This Summer deadlines:
  Homeworks and exams submission deadline: **9/04**
  Videos and waivers submission deadline: **9/11**

X-Camp Class Progress

skipping classes are encouraged



Invitation only

Competitive Class
竞赛
Fully committed to competitive programming
Grade: 7+/age: 12-15

Year 4+
(Year 3.5-4+)

Placement Test & Interview

CS601 **CS602**
- Extended Algorithms and Data Structure in C++

Targeting USACO Gold and above

CS603
Aim for entering the US Camp

Advanced
高级
Pass USACO Bronze
Grades: 6+/ages: 11-15

Year 3
(Year 2.5-3.5)

Placement Test

CS300 **CS301** **CS302** **CS402H**
- Coding structures and algorithms of C++

Targeting USACO Silver

USACO Booster Camps
Enhance competition experience
Every Winter & Summer

Intermediate
中级
At least 2-year coding experience
Grades: 6+/ages: 11-15

Year 2
(Year1.5-2.5)

Placement Test

CS200 **CS201** **CS202**
- Algorithms and Data Structure in C++

Targeting USACO Bronze

or

Robotics 2

APCS2 -- Java
1-year Java background required

Beginner
初级
No coding background to 1 year coding experience
Grades 6-8/age: 11-14

Year 1

Placement Test

CS100P **CS101P** **CS102P**
- Intro to Python Programming and Problem Solving

or

Robotics 1
1-year solid coding experience with Python, C++ or Java
Graders 6th - 12th

APCS 1 (6-weeks) -- Java
for students with some coding background
APCS Basics (10-weeks -- Java)
for students with no background
Graders 7th-9th

Contact us

Course Consultant

Wechat

Upgrade Qualification

Students need to be qualified for the next level study with the completion of at least 60% of required homework and the achievement of 60/100 corrected scores in the midterm test. (300 or above for this midterm. Please finish this in one week)

X-Camp USACO Achievements

- More than 100 X-Camp students have been qualified for USACO Silver division and above.
- 19 in the Platinum division and 5 selected in the US Camp, out of which 4 were fresh from the 2022 season.

X-Camp

X - Camp

Let's change the world with coding!

2022 Fall Weekend Classes

9/10-12/10 2022, 12 sessions

no class from 11/19-11/27

Programming with Python: CS100P-CS102P

No or half year coding experience | 6th+ Graders

Why Python python

Best programming language to enter the text-based coding world

Algorithms + Data Structures = Programs
AI with Python = Magic

From beginner level to intermediate level
Tuition Fees: $700-$1200

Classic C++ course: CS200 ~ CS402H

One year coding experience and above | 6th+ Graders

Why C++

Fundamental, adaptable, simpler, faster. The most preferred language for competitive programming.

Not only focus on language syntax, but Data Structure and Algorithms.

From Classic C++ to Competitional Honor Classes
Tuition Fees: $1200-$1400

USACO Grandmaster Classes CS601 ~ CS603*

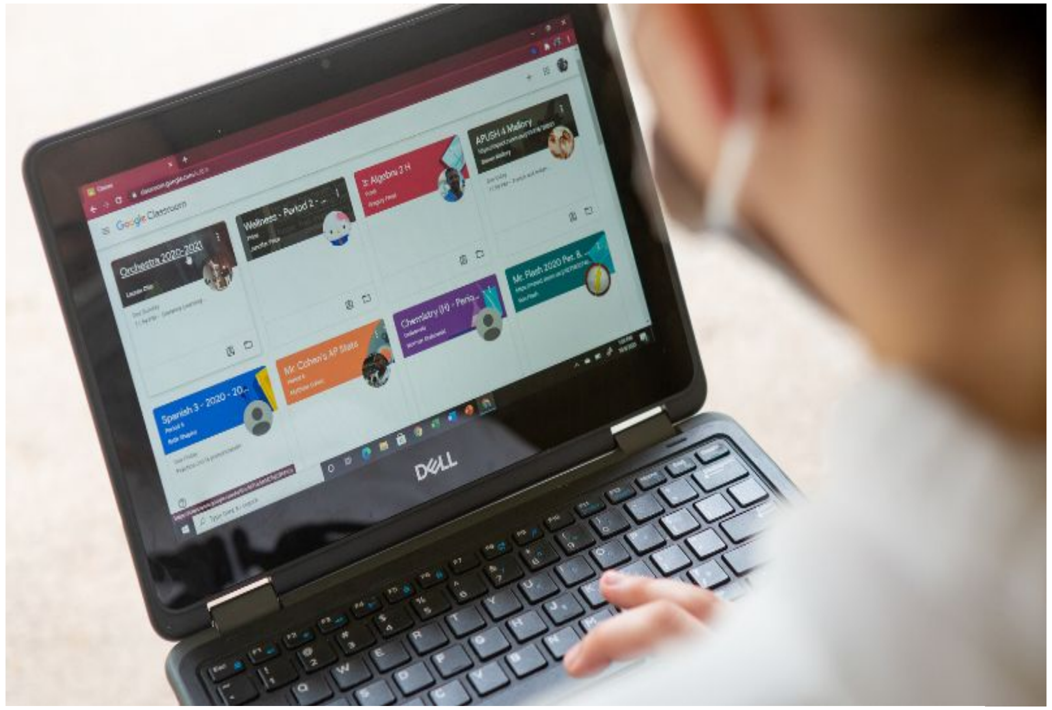
Passed USACO Silver | 7th+ Graders recommended

- **100% USACO Simulation Mock Test EVERY week, with selected problems and in-time analysis/group discussion**
- **Unsurpassed faculty including IOI Winners, ICPC World Finalist, and NOI Gold/Silver Medalist**
- **Dynamic class adjusting system, customized for students**
- **Strong CP community with 40+ top-level USACO students and faculties, brainstorming happens everyday**

Tuition Fees: $65/hour

- ➢ Please check your email from X-Camp or click the link in chat box to fill out the Midterm Parent Survey.

- ➢ **22 Fall Class Registration:** https://tinyurl.com/XCamp22Fall2



Early bird discount

$50 off by 8/1/2022

(payment date, not submission date)

http://x-camp.org

Register at

Contact us

Course Consultant

Wechat

X Camp 编程教育

X - Camp

P1005: Merge ordered lists

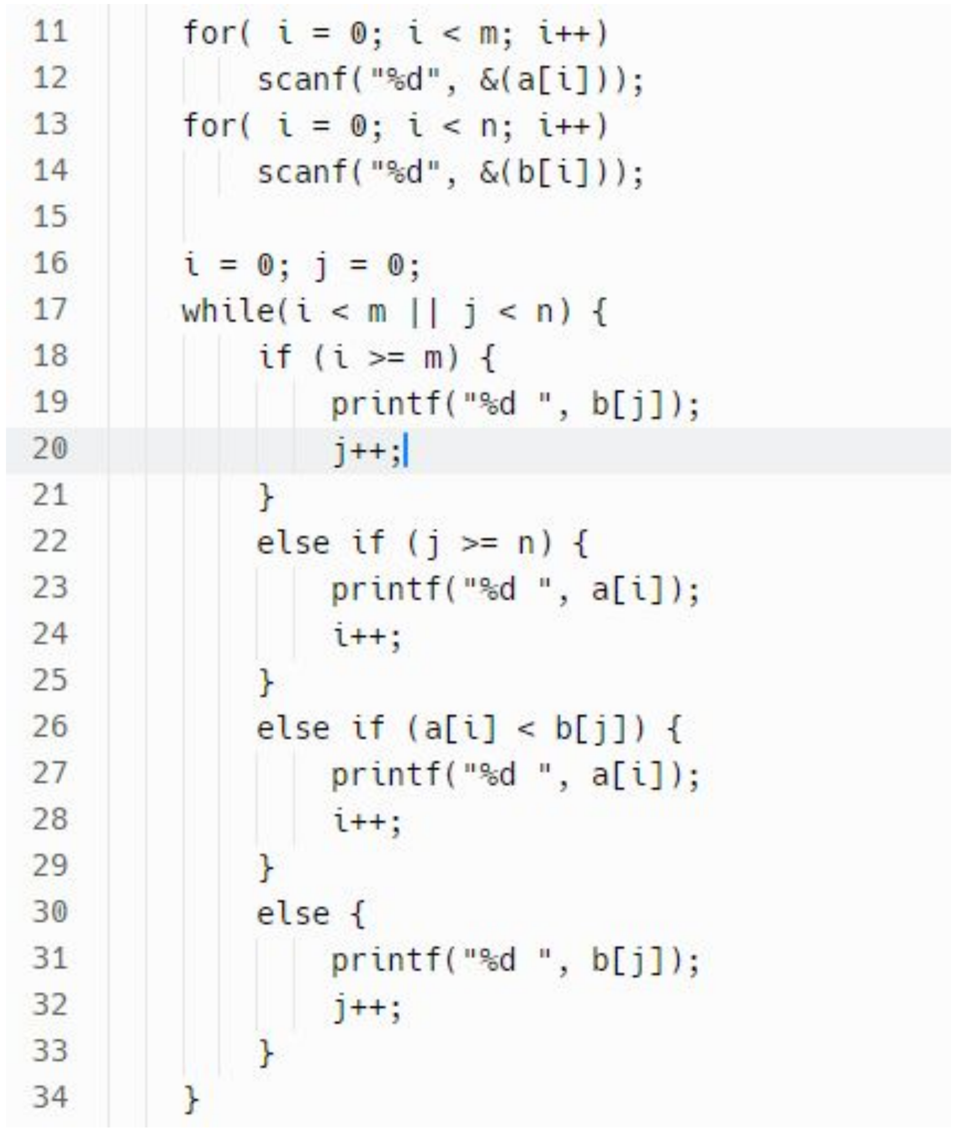
- We have 2 ordered list of integers
- Merge them in a new ordered list then output

- We can read the input then directly output

```
Sample input
4 5
1 4 7 8
2 4 6 9 10
```

```
sample output
1 2 4 4 6 7 8 9 10
```

P1005: Merge ordered lists



```
for( i = 0; i < m; i++)
    scanf("%d", &(a[i]));
for( i = 0; i < n; i++)
    scanf("%d", &(b[i]));

i = 0; j = 0;
while(i < m || j < n) {
    if (i >= m) {
        printf("%d ", b[j]);
        j++;
    }
    else if (j >= n) {
        printf("%d ", a[i]);
        i++;
    }
    else if (a[i] < b[j]) {
        printf("%d ", a[i]);
        i++;
    }
    else {
        printf("%d ", b[j]);
        j++;
    }
}
```

P7652: Machine Translation

- There is cache with specific size. Words will be moved from dictionary to cache until cache is full.
- If cache is full, oldest word will be removed from cache.
- How many moves to finish translation?

Sample input	sample output
3 7 1 2 1 5 4 4 1	5

- We need a way to store words in cache and quickly check existence

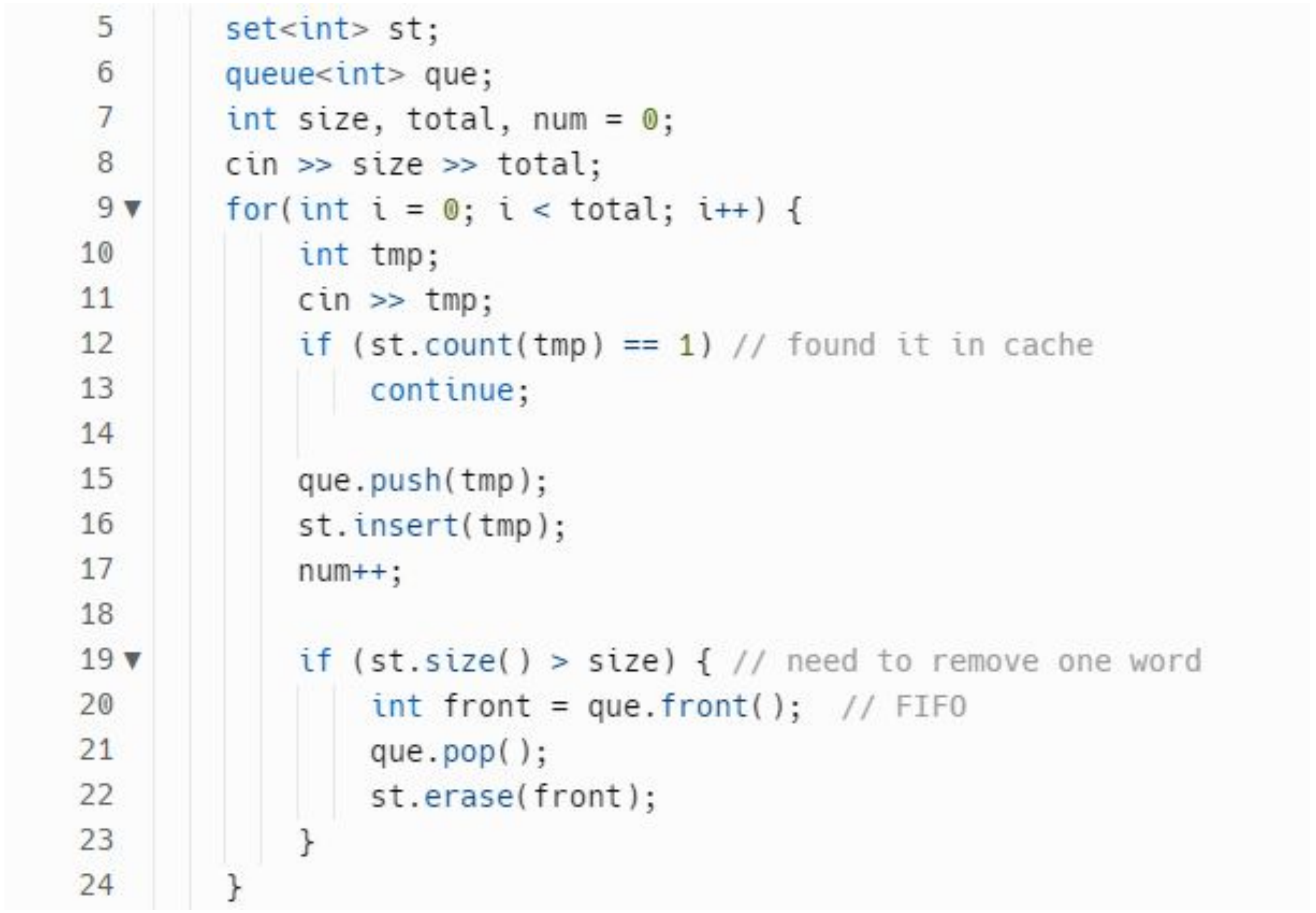
P7652: Machine Translation

- There is cache with specific size. Words will be moved from dictionary to cache until cache is full.
- If cache is full, oldest word will be removed from cache.
- How many moves to finish translation?

Sample input	sample output
3 7	5
1 2 1 5 4 4 1	

- We need a way to store words in cache and quickly check existence

P7652: Machine Translation



```
set<int> st;
queue<int> que;
int size, total, num = 0;
cin >> size >> total;
for(int i = 0; i < total; i++) {
    int tmp;
    cin >> tmp;
    if (st.count(tmp) == 1) // found it in cache
        continue;

    que.push(tmp);
    st.insert(tmp);
    num++;

    if (st.size() > size) { // need to remove one word
        int front = que.front();  // FIFO
        que.pop();
        st.erase(front);
    }
}
```

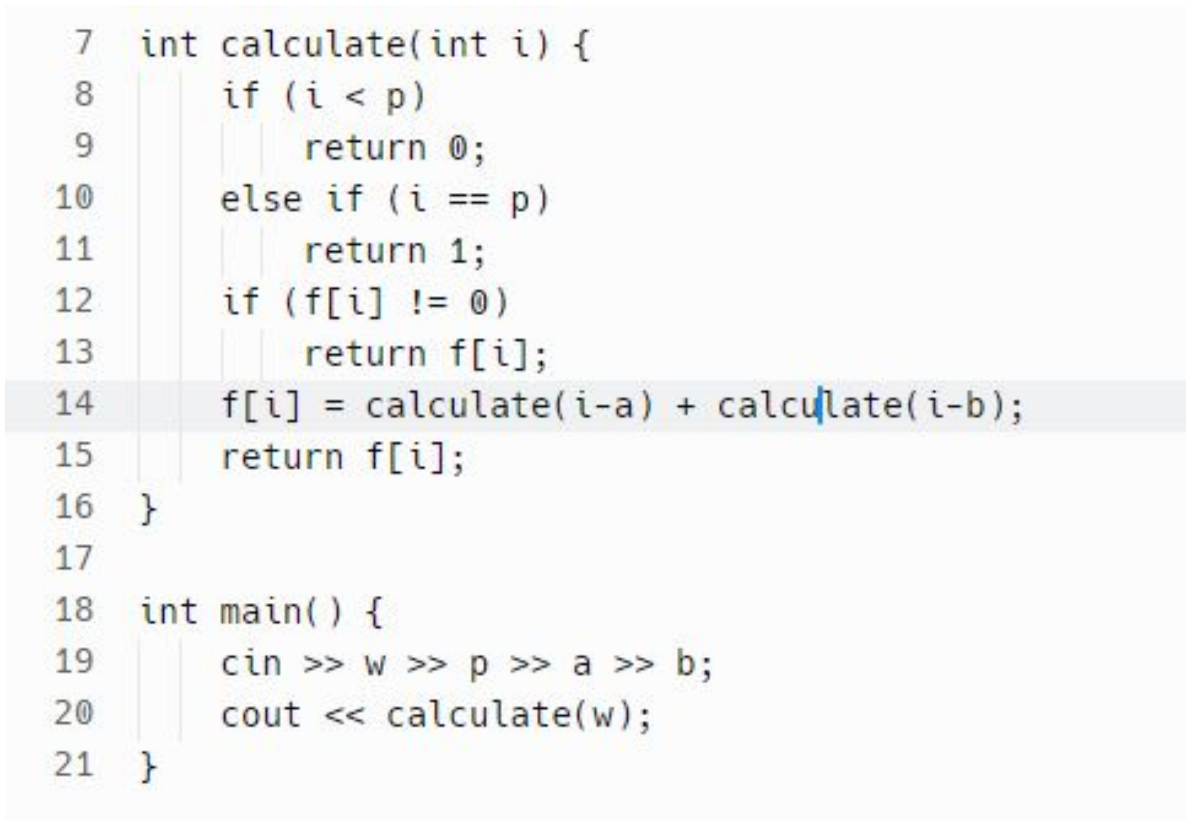
P8468: A Strange Auction

- How many ways from w to p?
- Every time, you can move a or b

Sample input	sample output
10 3 2 3	3

- Solution for sample data is: 10 - 8 - 6 - 3; 10 - 8 - 5 - 3; 10 - 7 - 5 - 3
- Typical DFS/BFS? You may have TLE error
- Remember our old recursion algorithm?

P8468: A Strange Auction



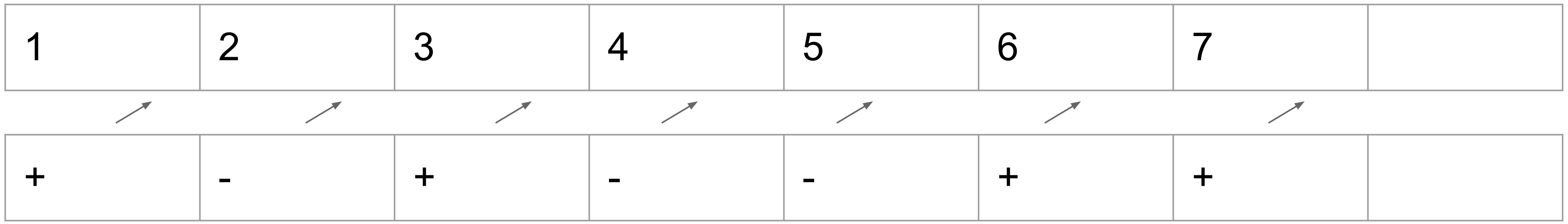
```
int calculate(int i) {
    if (i < p)
        return 0;
    else if (i == p)
        return 1;
    if (f[i] != 0)
        return f[i];
    f[i] = calculate(i-a) + calculate(i-b);
    return f[i];
}

int main() {
    cin >> w >> p >> a >> b;
    cout << calculate(w);
}
```

P8018: Number sequence

- Sequence of digits from 1 to N
- Add ‘ ’, ‘+’ or ‘-’ between them to make calculation result to be 0
- In the increasing order of ASCII code (' ' < '+' < '-')
- How many ways?
- DFS or BFS?

Sample input	sample output
7	1+2-3+4-5-6+7
	1+2-3-4+5+6-7
	1-2 3+4+5+6+7
	1-2 3-4 5+6 7
	1-2+3+4-5+6-7
	1-2-3-4-5+6+7

P8018: Number sequence

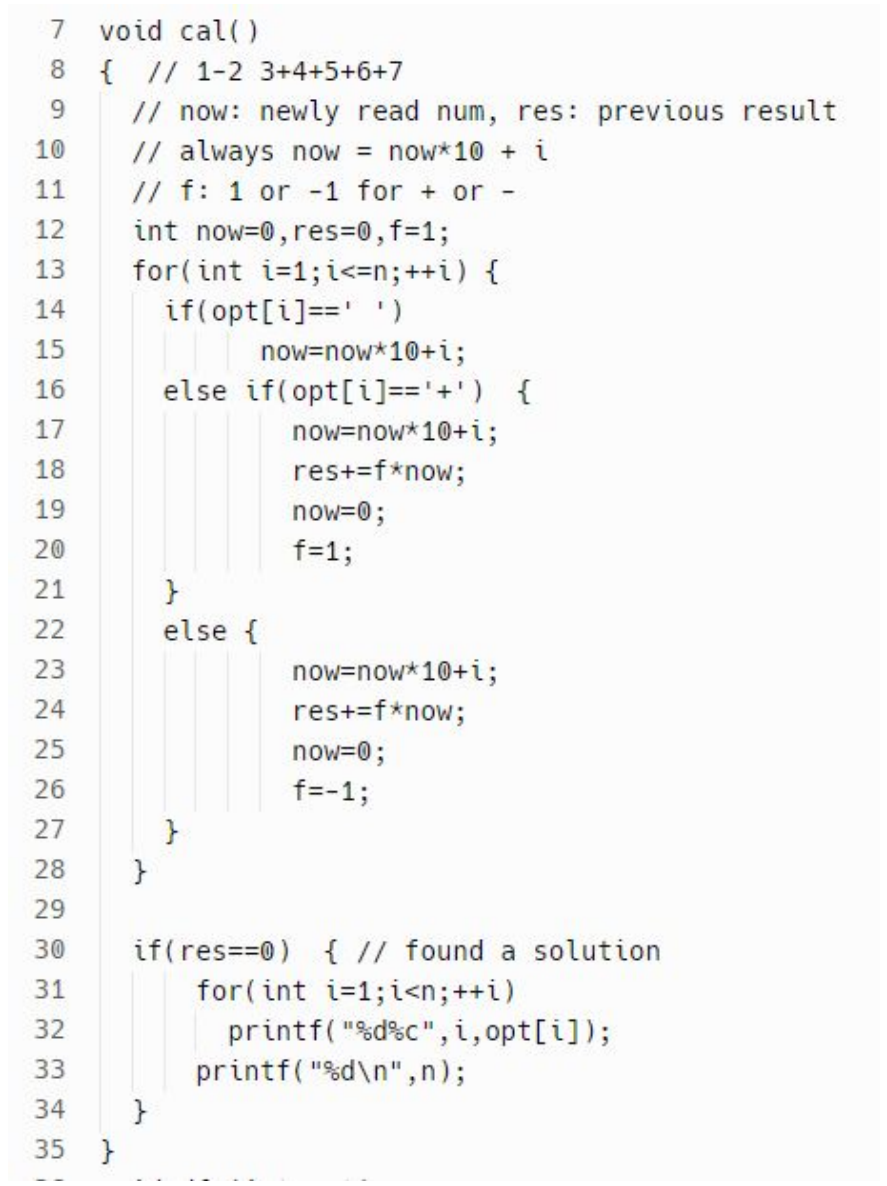


- 1+2-3+4-5-6+7
- Use a operator array to store operators
- Need to have a function to calculate expression

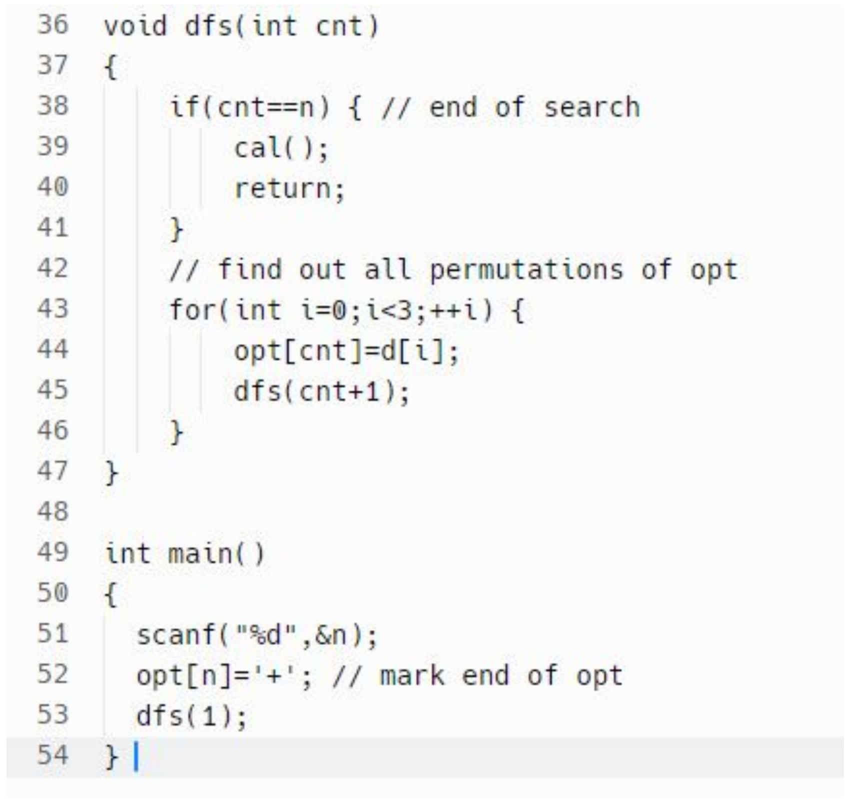
State	char d[]={" +-"},opt[10];	

Transition	3 states	for(int i=0;i<3;++i) {
		    opt[cnt]=d[i];
		    dfs(cnt+1);
		}

P8018: Number sequence



```
void cal()
{  // 1-2 3+4+5+6+7
 // now: newly read num, res: previous result
 // always now = now*10 + i
 // f: 1 or -1 for + or -
 int now=0,res=0,f=1;
 for(int i=1;i<=n;++i) {
  if(opt[i]==' ')
     now=now*10+i;
  else if(opt[i]=='+')  {
      now=now*10+i;
      res+=f*now;
      now=0;
      f=1;
  }
  else {
      now=now*10+i;
      res+=f*now;
      now=0;
      f=-1;
  }
 }

 if(res==0)  { // found a solution
   for(int i=1;i<n;++i)
    printf("%d%c",i,opt[i]);
   printf("%d\n",n);
 }
}
```



```
void dfs(int cnt)
{
  if(cnt==n) { // end of search
    cal();
    return;
  }
  // find out all permutations of opt
  for(int i=0;i<3;++i) {
    opt[cnt]=d[i];
    dfs(cnt+1);
  }
}

int main()
{
 scanf("%d",&n);
 opt[n]='+'; // mark end of opt
 dfs(1);
}
```

P7420: Shortest Path of Pawn

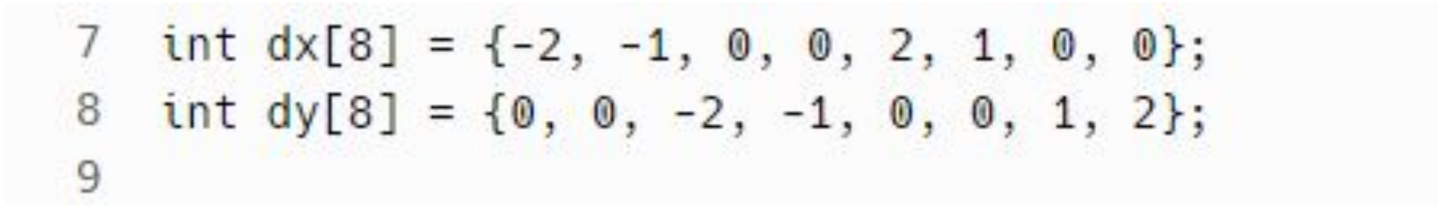
- Least number of steps a pawn moves from one position to another
- move north, south, east or west with either 1 or 2 steps
- No blockers with start and end points
- DFS or BFS or something else?
- Many solutions. We can take BFS as example

Sample input	sample output

100
 	
0 0
 	
1 1
	2

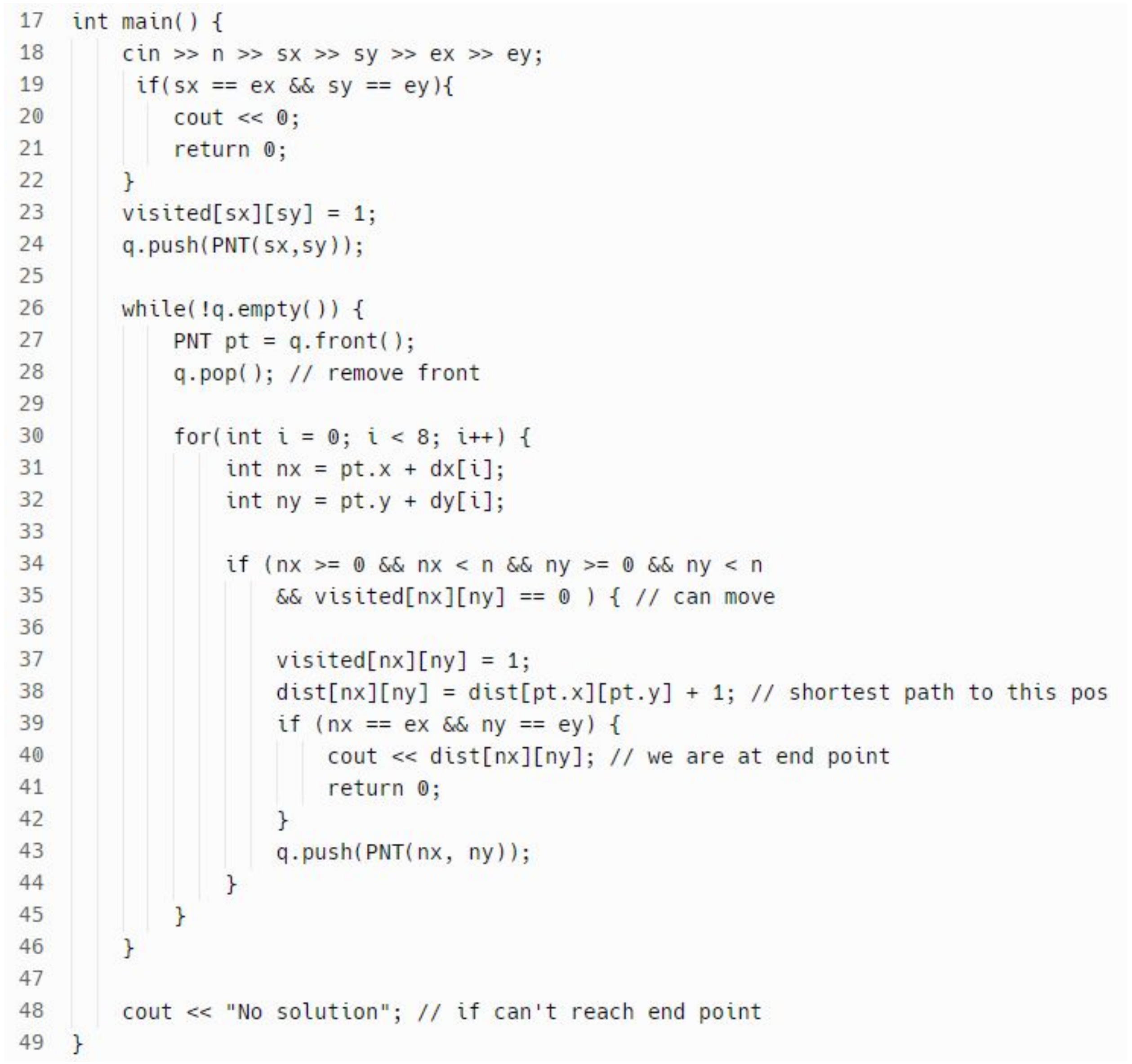
P7420: Shortest Path of Pawn

- Let’s review our basic algorithm for BFS
- In class practice https://onlinegdb.com/j_57qpvdD
- Very similar to this sample code for BFS
- No blockers, only need to check in-bound
- Pawn can move either 1 or 2 steps for each move
- Need to modify dx[], dy[]



```
int dx[8] = {-2, -1, 0, 0, 2, 1, 0, 0};
int dy[8] = {0, 0, -2, -1, 0, 0, 1, 2};

```

P7420: Shortest Path of Pawn



```
int main() {
    cin >> n >> sx >> sy >> ex >> ey;
     if(sx == ex && sy == ey){
        cout << 0;
        return 0;
    }
    visited[sx][sy] = 1;
    q.push(PNT(sx,sy));

    while(!q.empty()) {
        PNT pt = q.front();
        q.pop(); // remove front

        for(int i = 0; i < 8; i++) {
            int nx = pt.x + dx[i];
            int ny = pt.y + dy[i];

            if (nx >= 0 && nx < n && ny >= 0 && ny < n
                && visited[nx][ny] == 0 ) { // can move

                visited[nx][ny] = 1;
                dist[nx][ny] = dist[pt.x][pt.y] + 1; // shortest path to this pos
                if (nx == ex && ny == ey) {
                    cout << dist[nx][ny]; // we are at end point
                    return 0;
                }
                q.push(PNT(nx, ny));
            }
        }
    }

    cout << "No solution"; // if can't reach end point
}
```

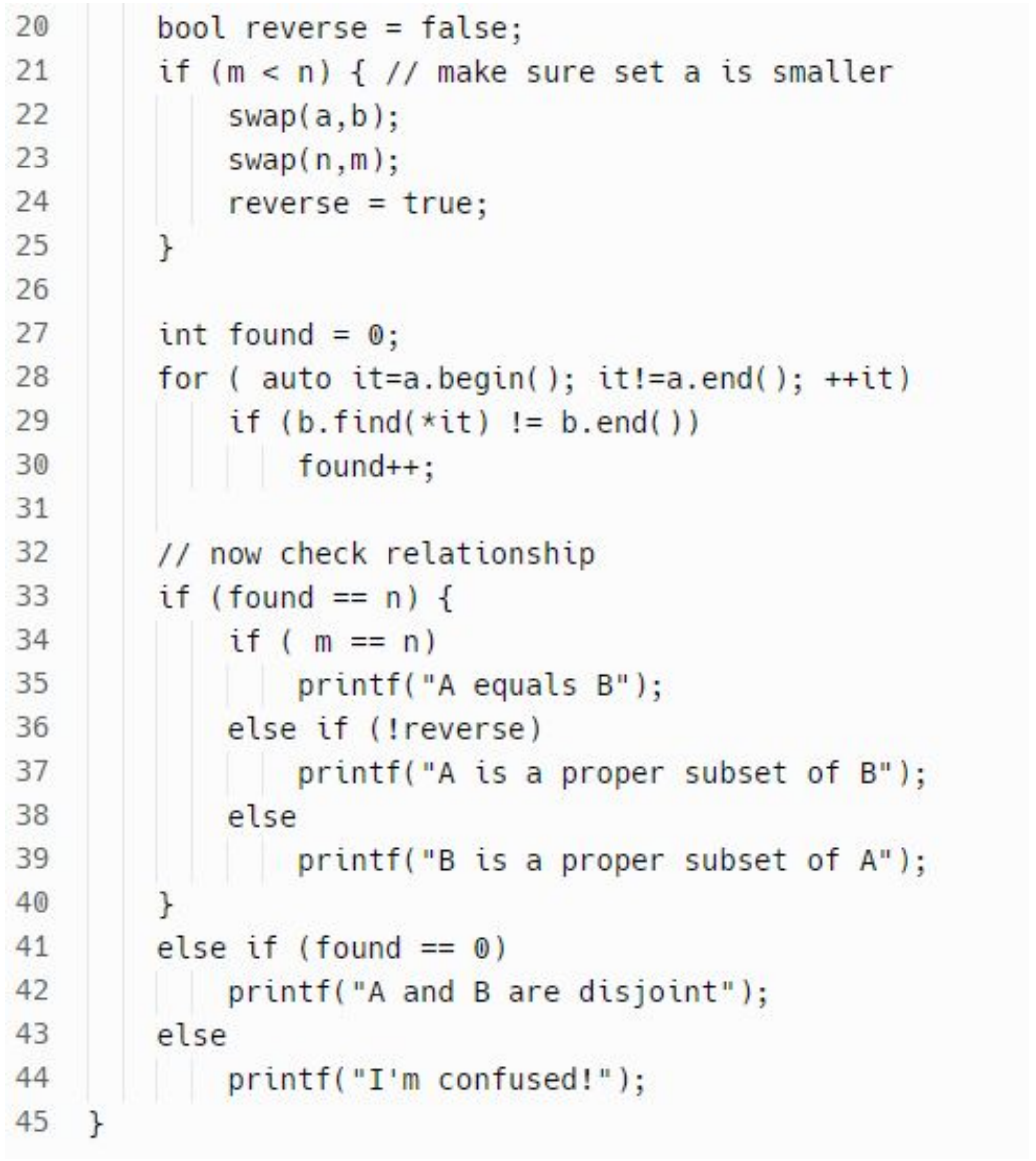
HW5: P1039: Set

Sample Input:

```
5 3
2 4 6 7 8
2 6 7
```

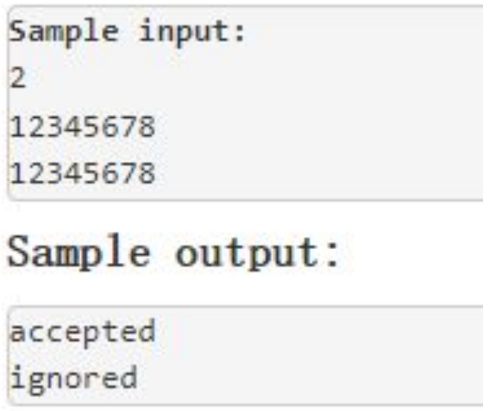
Sample Output:

```
B is a proper subset of A
```



```
    bool reverse = false;
    if (m < n) { // make sure set a is smaller
        swap(a,b);
        swap(n,m);
        reverse = true;
    }

    int found = 0;
    for ( auto it=a.begin(); it!=a.end(); ++it)
        if (b.find(*it) != b.end())
            found++;

    // now check relationship
    if (found == n) {
        if ( m == n)
            printf("A equals B");
        else if (!reverse)
            printf("A is a proper subset of B");
        else
            printf("B is a proper subset of A");
    }
    else if (found == 0)
        printf("A and B are disjoint");
    else
        printf("I'm confused!");
}
```

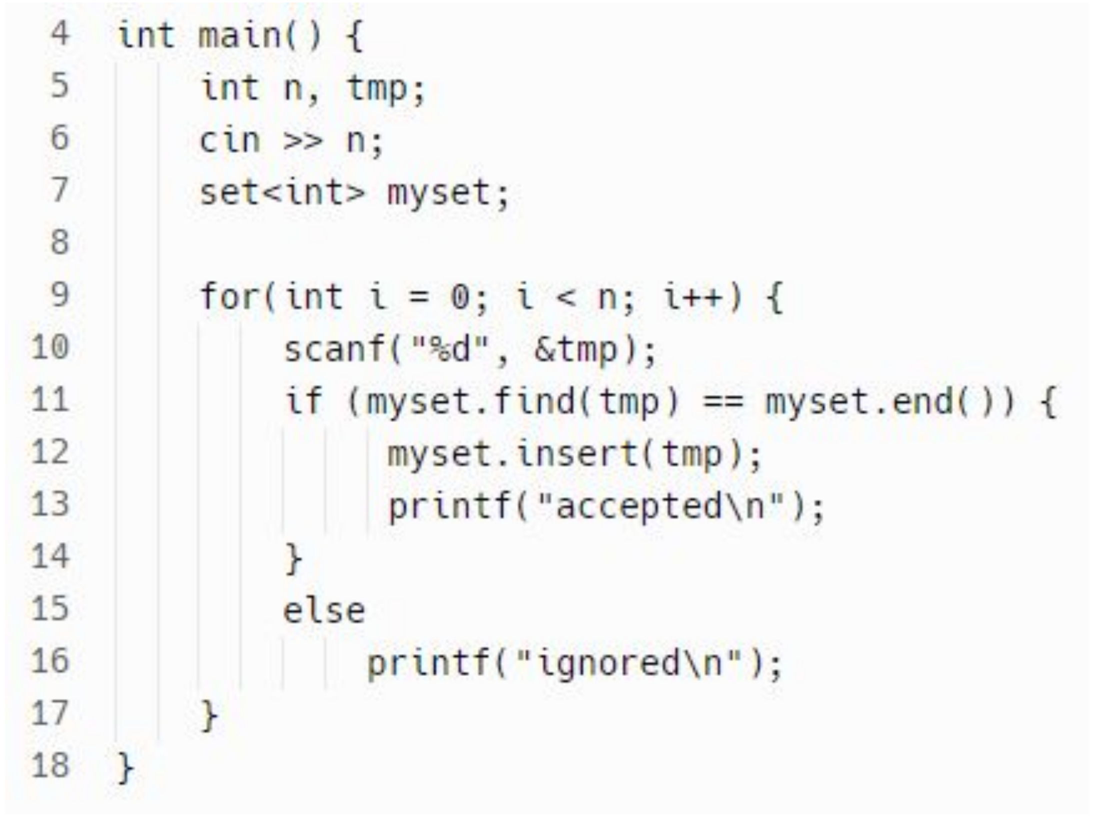
HW5: P3614: Integers Search



```
Sample input:
2
12345678
12345678
```

Sample output:

```
accepted
ignored
```



```
int main() {
    int n, tmp;
    cin >> n;
    set<int> myset;

    for(int i = 0; i < n; i++) {
        scanf("%d", &tmp);
        if (myset.find(tmp) == myset.end()) {
            myset.insert(tmp);
            printf("accepted\n");
        }
        else
            printf("ignored\n");
    }
}
```

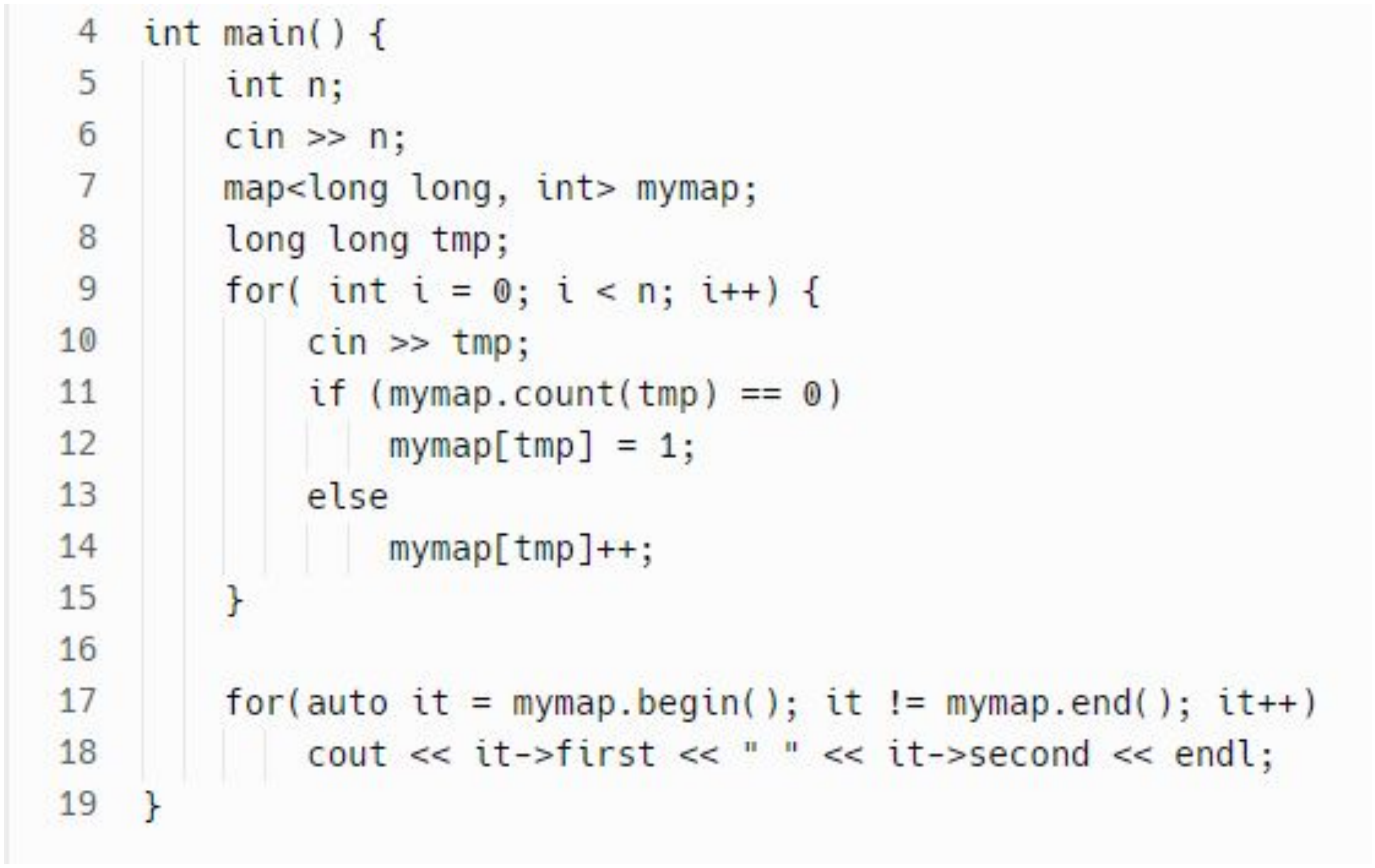
HW5: P1040: Counting

Sample Input:

```
9 (the total number of integer n)
2

4

6

2

6

4

2

8

6
```

Sample output:

```
2 3
4 2
6 3
8 1
```



```cpp
int main() {
    int n;
    cin >> n;
    map<long long, int> mymap;
    long long tmp;
    for( int i = 0; i < n; i++) {
        cin >> tmp;
        if (mymap.count(tmp) == 0)
            mymap[tmp] = 1;
        else
            mymap[tmp]++;
    }

    for(auto it = mymap.begin(); it != mymap.end(); it++)
        cout << it->first << " " << it->second << endl;
}
```

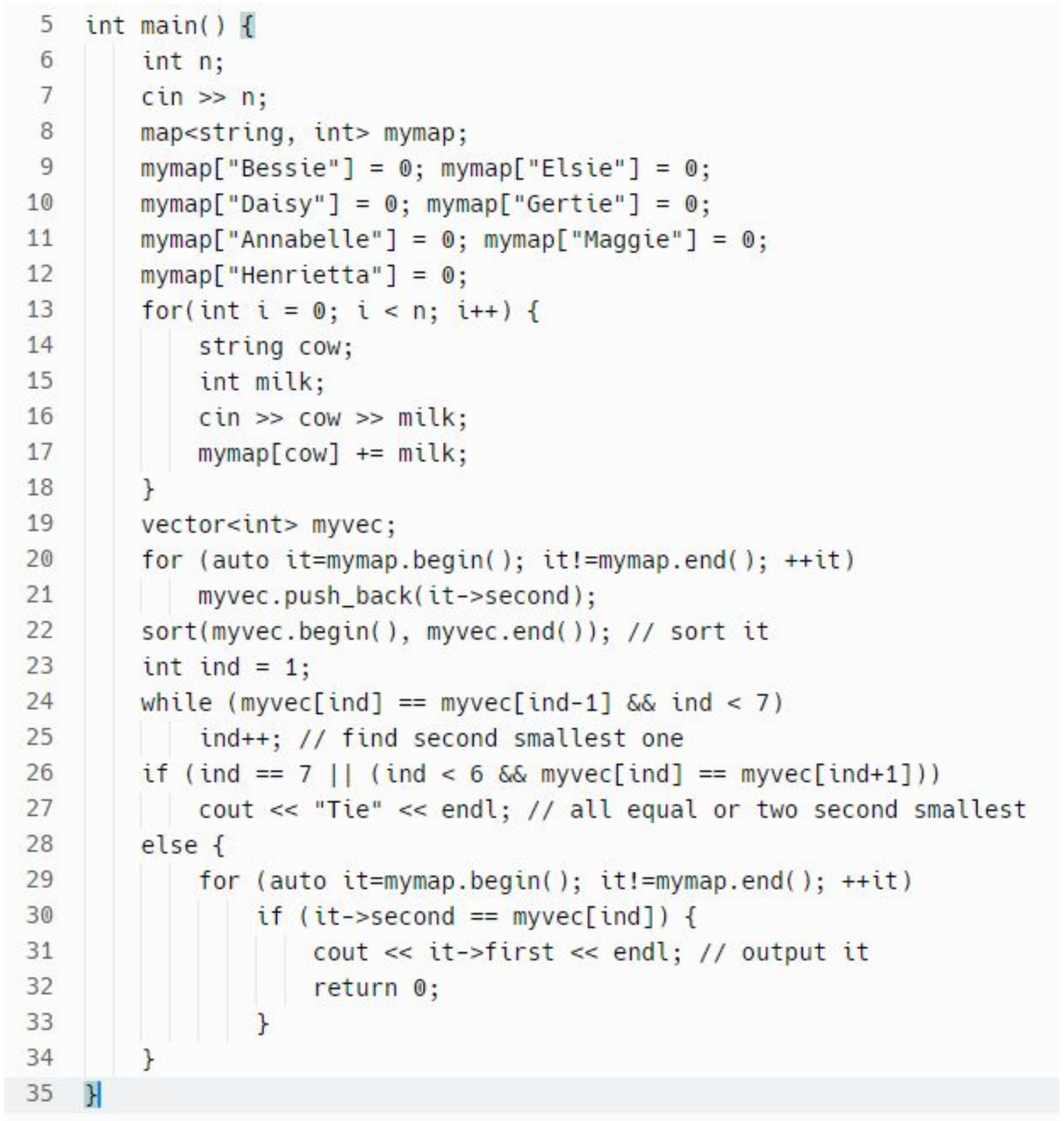
HW5: 4068: Don't be last

SAMPLE INPUT:

```
10
Bessie 1
Maggie 13
Elsie 3
Elsie 4
Henrietta 4
Gertie 12
Daisy 7
Annabelle 10
Bessie 6
Henrietta 5
```

SAMPLE OUTPUT:

```
Henrietta
```



```cpp
int main() {
    int n;
    cin >> n;
    map<string, int> mymap;
    mymap["Bessie"] = 0; mymap["Elsie"] = 0;
    mymap["Daisy"] = 0; mymap["Gertie"] = 0;
    mymap["Annabelle"] = 0; mymap["Maggie"] = 0;
    mymap["Henrietta"] = 0;
    for(int i = 0; i < n; i++) {
        string cow;
        int milk;
        cin >> cow >> milk;
        mymap[cow] += milk;
    }
    vector<int> myvec;
    for (auto it=mymap.begin(); it!=mymap.end(); ++it)
        myvec.push_back(it->second);
    sort(myvec.begin(), myvec.end()); // sort it
    int ind = 1;
    while (myvec[ind] == myvec[ind-1] && ind < 7)
        ind++; // find second smallest one
    if (ind == 7 || (ind < 6 && myvec[ind] == myvec[ind+1]))
        cout << "Tie" << endl; // all equal or two second smallest
    else {
        for (auto it=mymap.begin(); it!=mymap.end(); ++it)
            if (it->second == myvec[ind]) {
                cout << it->first << endl; // output it
                return 0;
            }
    }
}
```

Review classic BFS implementation

Define struct Node and store different things inside it to achieve different goals.

```
queue<Node> q;
While (q not empty) {
    Pop head from q
    Expand next layer from the head and push them into the q
    If reach destination, handle it by increment counter or print something
}
```

BFS - find distance of the shortest path

Procedure - how to do it bfs (sx, sy)

1. Add the start node to the queue
2. While queue is not empty
   a. Take the front node out of the queue
   b. Visit all neighbors from this node
      i. Set the distance (current dist + 1)
      ii. If it’s the end node, done.
      iii. Mark visited true
      iv. Push the new node to the queue

```
q.push(sx,sy);
while (!q.empty()) {
  (x, y) = q.front();
  for (all the directions) {
    nx = x + dx[i]; //new location
    ny = y + dy[i];
    if (new location not valid) continue;
    dist[nx][ny] = dist[x][y]+1;

    if( (nx, ny) is end point) { done. }

    visited[nx][ny] = 1;
    q.push(point(nx, ny));
  }
}
cout << dist[ex][ey]
```

Pioneer Problem P8119

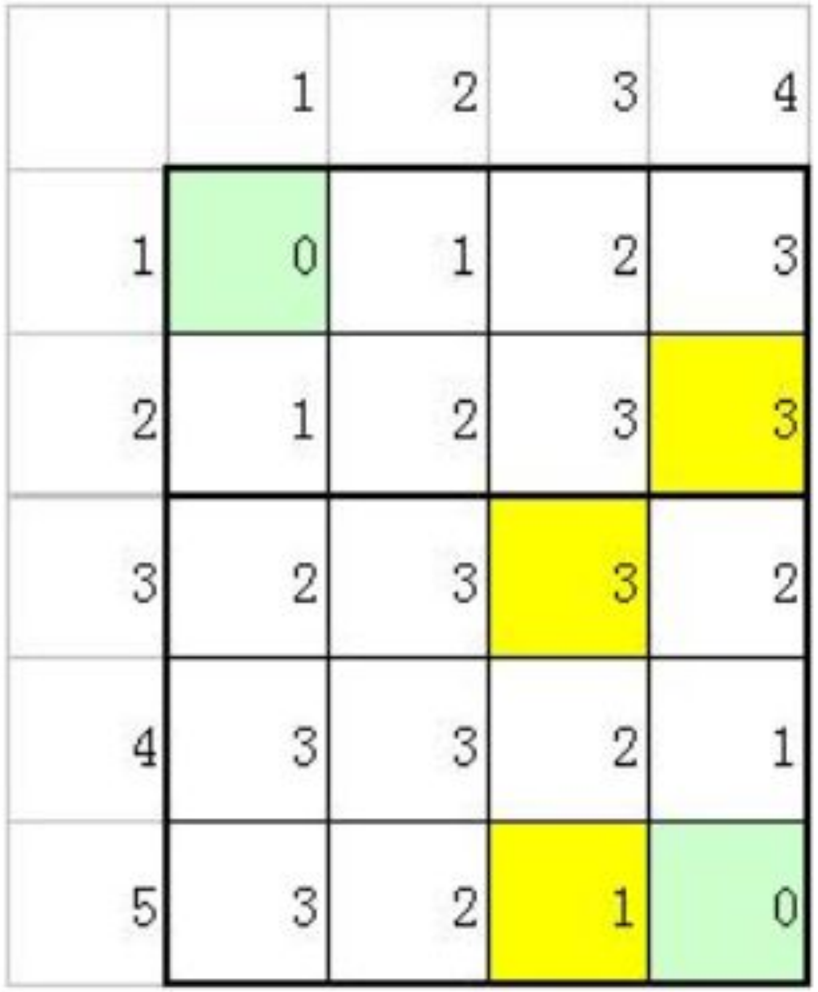
https://xjoi.net/contest/4643 problem 2

Sample input 1:

```
5 4 2 3
1 1
5 4
3 3
5 3
2 4
```

Sample output 1:

```
3
1
3
```



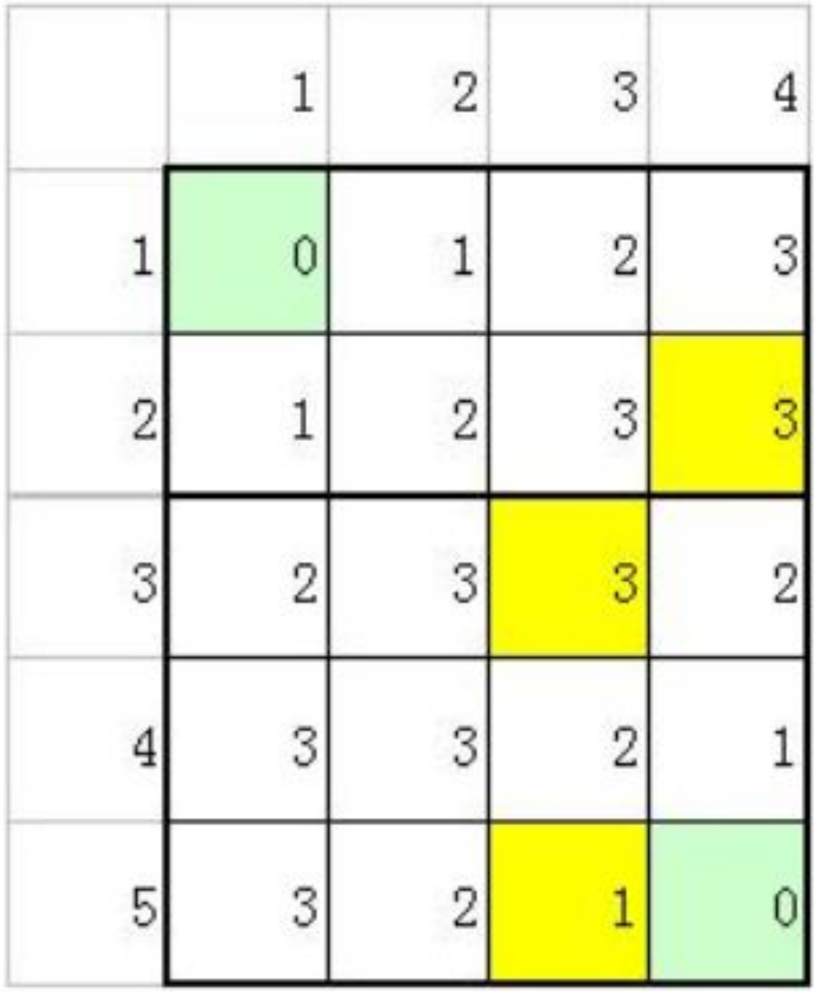
	1	2	3	4
1	🟩 0	1	2	3
2	1	2	3	🟨 3
3	2	3	🟨 3	2
4	3	3	2	1
5	3	2	🟨 1	🟩 0

Pioneer Problem

Multiple sources

If one position was reached earlier by a source, when another source is coming, we don’t add this position into queue

Because it is not optimal



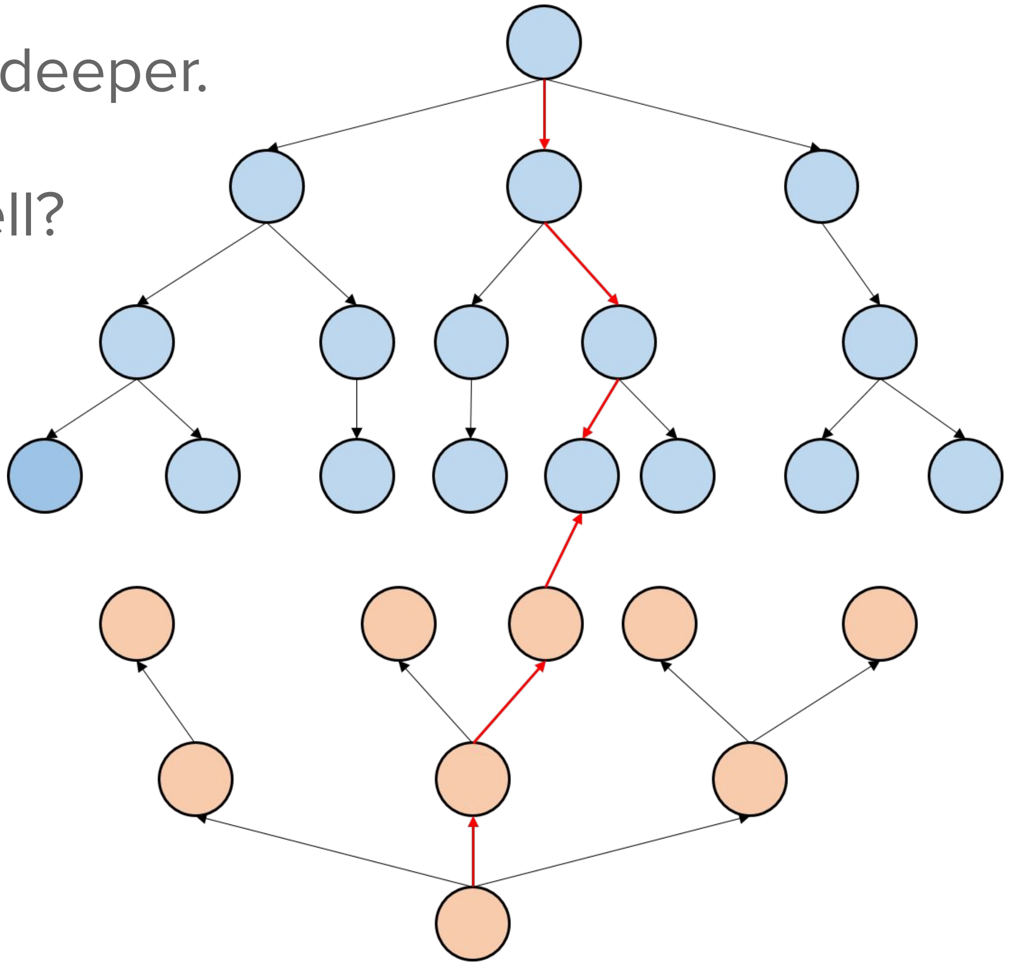
	1	2	3	4
1	0	1	2	3
2	1	2	3	3
3	2	3	3	2
4	3	3	2	1
5	3	2	1	0

How can we make BFS faster?

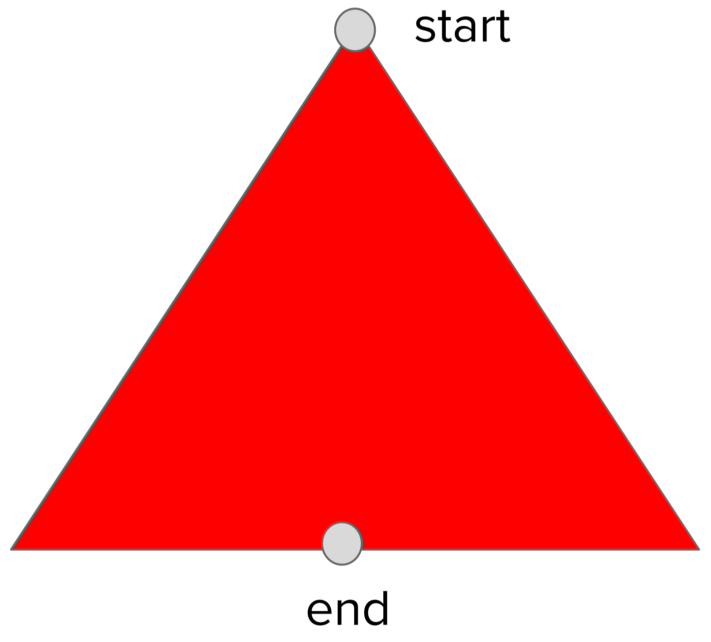
States in one layer increases over time as we go deeper.

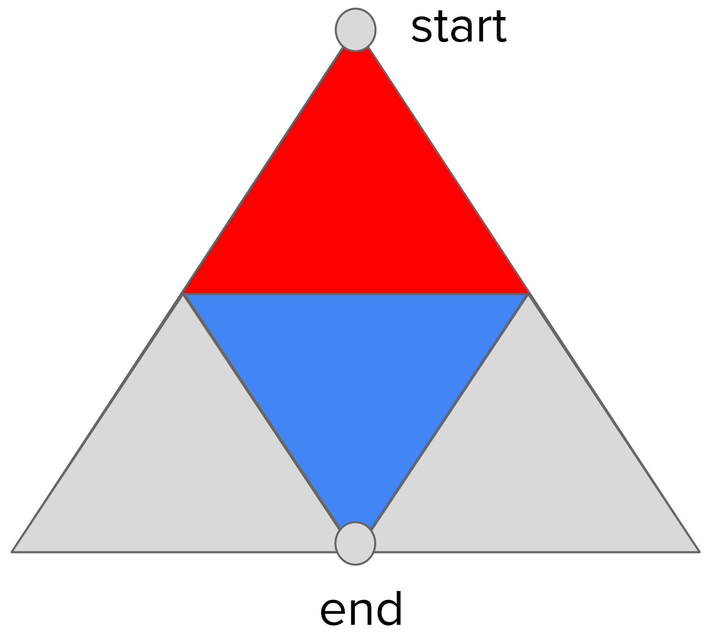
Can we expand from the exit layer by layer as well?

How would this change the performance?



Bi-directional BFS





BFS searches all states.

Bidirectional BFS.

Red part is forward direction BFS.

Blue part is backward direction BFS.

Gray part saved by bidirectional BFS.

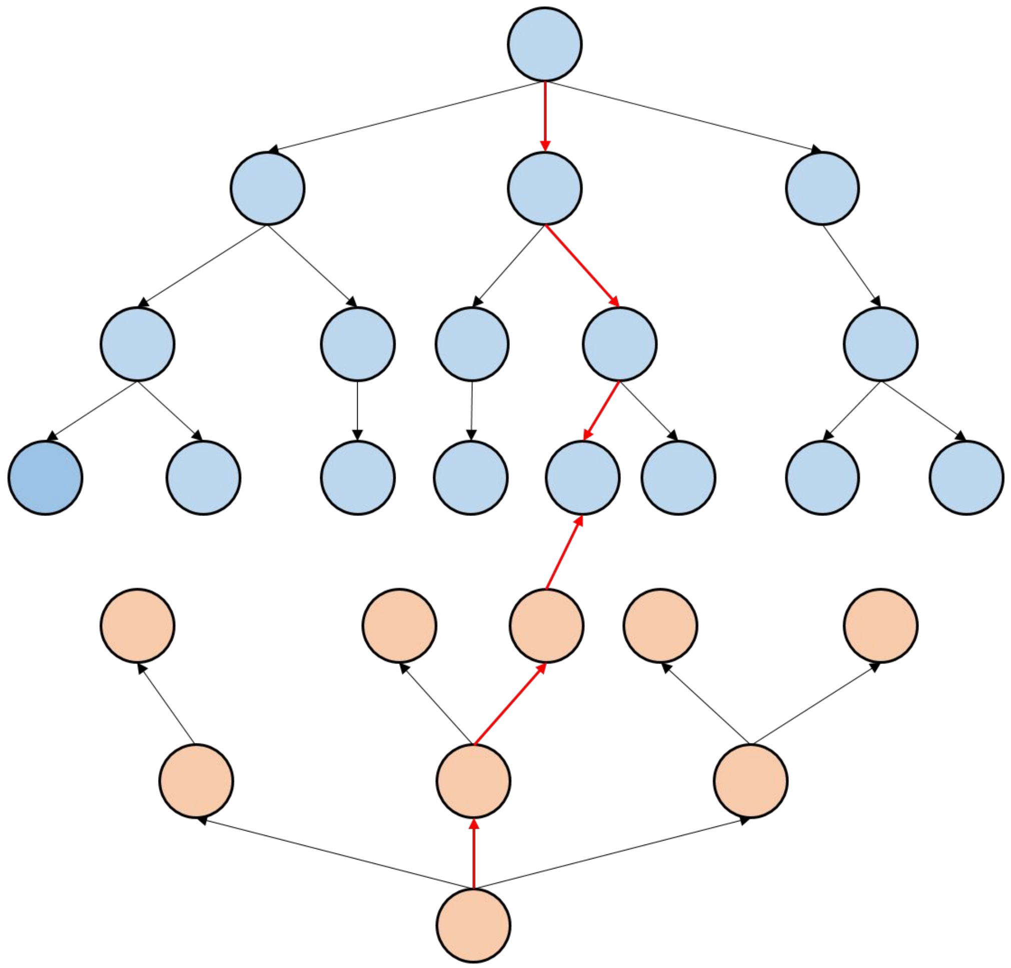
How to do bidirectional BFS?

Think about how the process works.

Develop one layer in each direction for each iteration.

If we can find out one path connecting forward BFS states and backward BFS states, we find the solution.

Check for connection for each newly expanded layer of states.

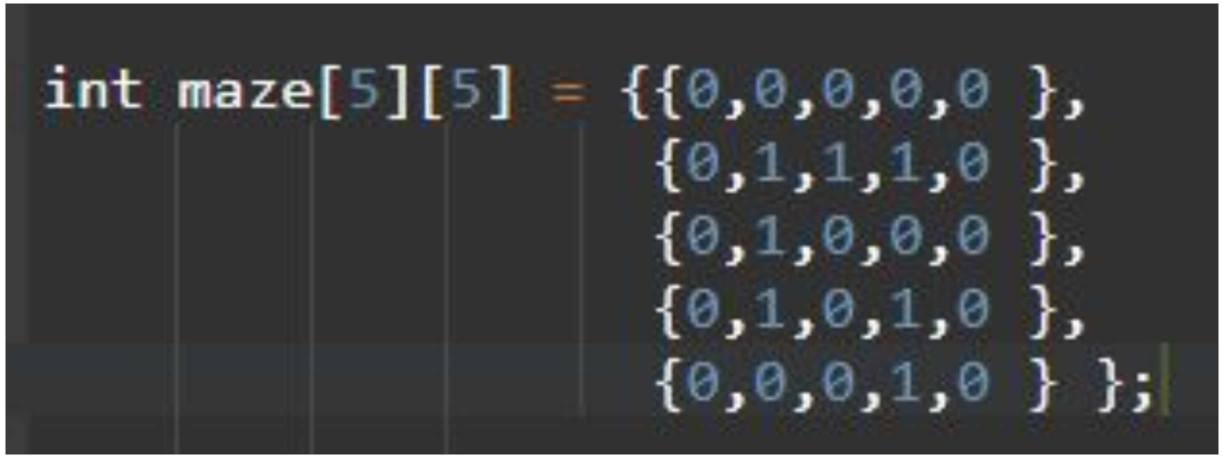


Using bidirectional BFS

In this example, how do we implement bidirectional BFS?

Starting point (0,0)

End point (4, 4)



```
int maze[5][5] = {{0,0,0,0,0 },
                   {0,1,1,1,0 },
                   {0,1,0,0,0 },
                   {0,1,0,1,0 },
                   {0,0,0,1,0 } };
```

```
queue<PNT> q, rq;
map<PNT, step> step, rstep;
```

Word Ladder (P12608)

You have a dictionary with N words. There're one start word and one target word. For each step, you can choose to delete, add, or modify one char in your existing word, to form a new word if the new word exist in the dictionary. Find out min steps required to go from start word to target word.

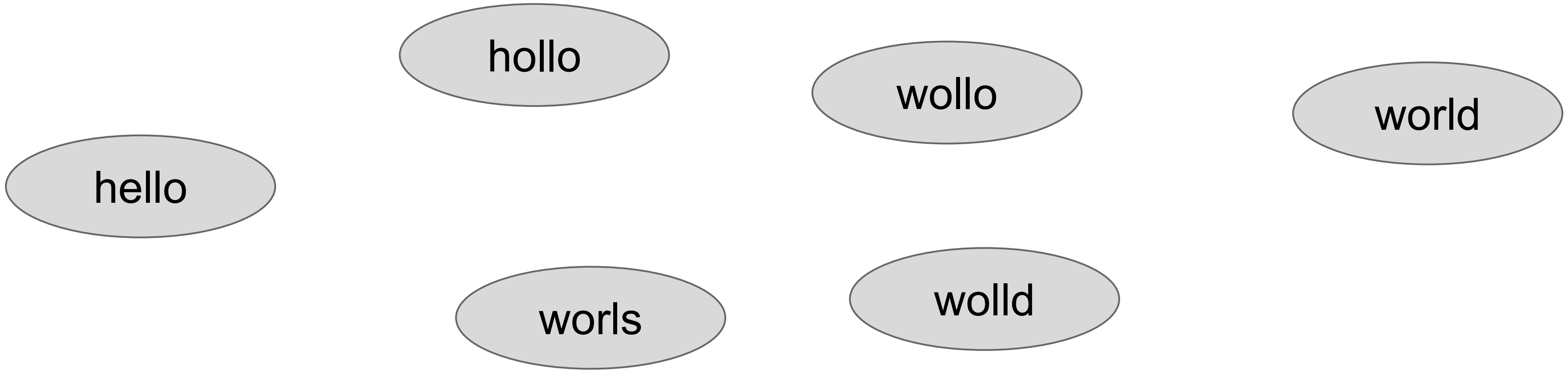
Input:
```
hello    // start
world    // target
6
hello
hollo
worls
wollo
wolld
world
```

Output:
```
4
```

https://xjoi.net/contest/4643 problem 3

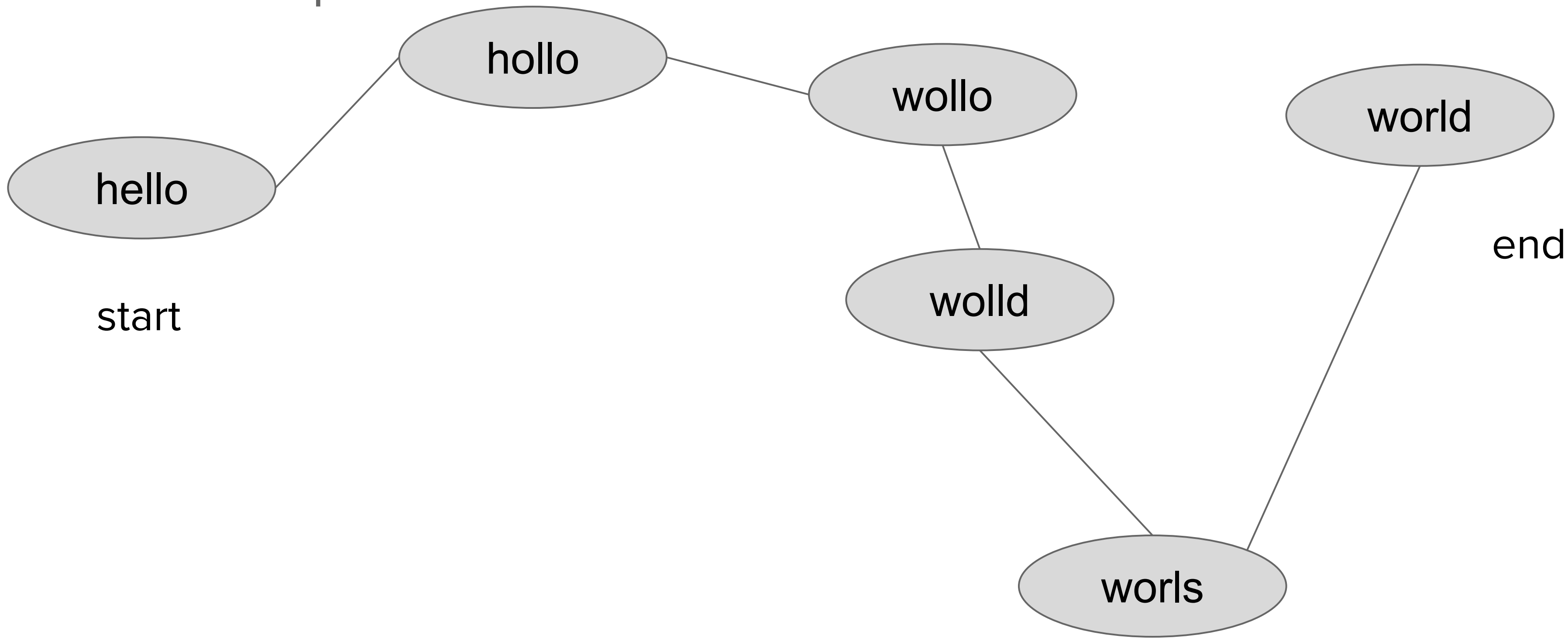
Word Ladder

Treat each of the word as one node/state. Add an edge to connect two words that are one transition away.



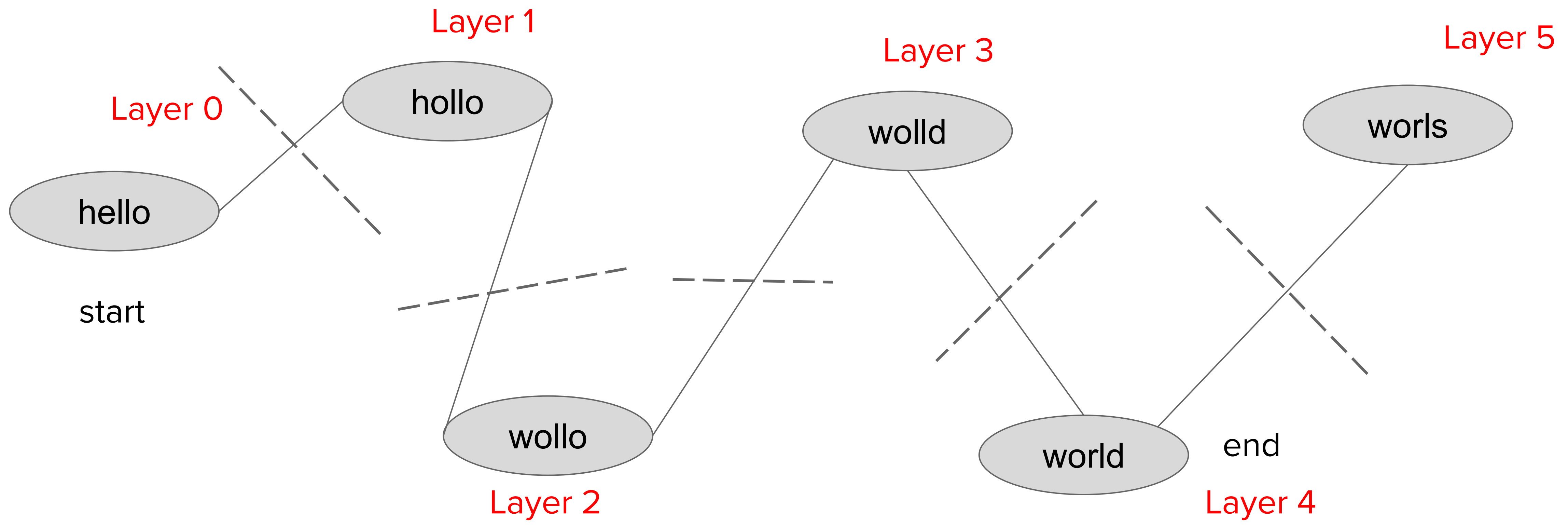
Word Ladder

Problem transformed into - find the shortest path from start point to the destination point.



Build a ladder from start to end

BFS - search layer by layer. For each queue head, enqueue all the possible words that it can get to with one step. Terminate if we reach destination word.



For simplicity, there are only one node for each layer in this example.

BFS Pseudocodes for word ladder

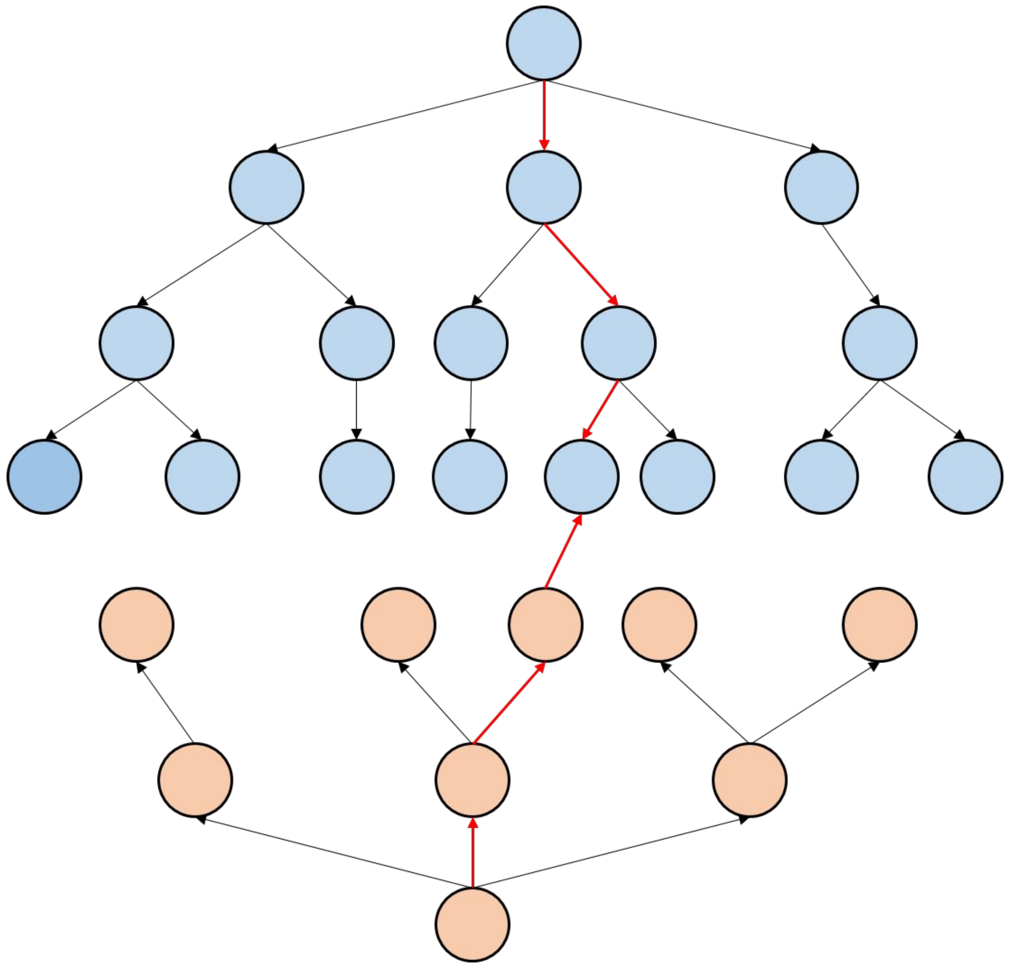
```
queue<node> q;
q.push(start);
while (!q.empty()) {
    head = q.front();
    q.pop();
    Enqueue all words that head can get to with one step (one modification)
    Termination check
    Infinity circle check
}
```

State Tree

BFS is to go layer by layer from one start word to approach the destination word.

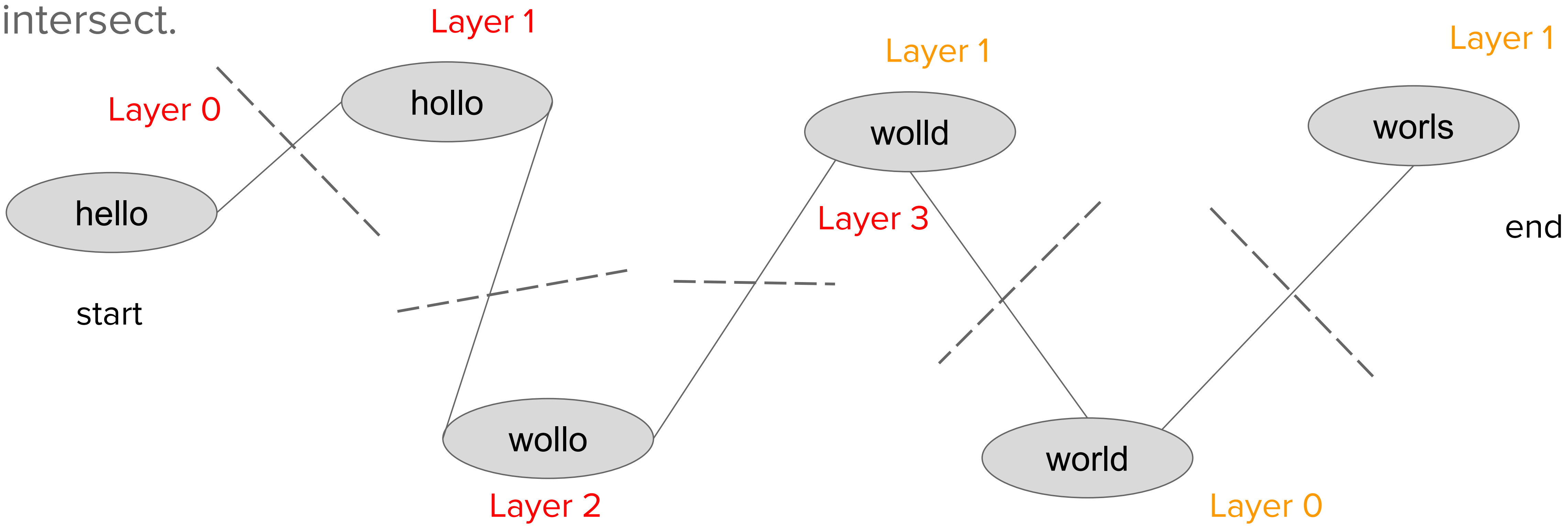
What if we go both direction?

How would this influence the performance?



Use Bi-directional BFS

BFS from both start word and target word. Use two queues to maintain the steps from start word and the steps from target word, respectively. Terminate if they intersect.



Bi-directional BFS for word ladder

```
queue<node> q, rq;
q.push(start);     rq.push(end);
while (!q.empty() && !rq.empty()) {
        If (!q.empty()) {
                head = q.front();      q.pop();
                Enqueue all words that head can get to with one step (one modification)
                Termination check (whether the word has appeared in rq)
                Infinity circle check
        }

If (!rq.empty()) {
                head = rq.front();                rq.pop();
                Enqueue all words that head can get to with one step (one modification)
                Termination check (whether the word has appeared in q)
                Infinity circle check
        }
}
```

Word Ladder

BFS - for each iteration, expand to all the words that current word can.

Problem 12608, also HW this week

Question: how do we check if two words are neighbors (move from a state to another state)?

How do we know hello and hollo are neighbors?

How about helloo? hellooo? hell? hel?

Word Ladder - check two words

- For word in dictionary, check if it is neighbor of current word
- check lengths, if difference > 1, false
- if difference == 1
  - remove element from longer string to see if they match
- if same length
  - check how many elements are different
  - diff == 1 → yes; otherwise → no