X-Camp

X-CAMP

2022 Summer Quarter

CS202B / Nick Wang

The Teaching Team

- Teacher - Nick Wang, 15 years professional experience on software development, 2+ years teaching experiences in X-Camp.

- TA - Peter
- TA - Alex
- TA - Cory

- Peter: Wednesday 7-8PM
- Alex: Thursday 7-8PM
- Cory: Friday 7-8PM

Syllabus

X-Camp CS202 class will continue to enforce the learning of DFS and BFS algorithms for solving more complicated problems, and introduce more data structures and algorithms. Then we train the students how to apply these algorithms to solve many different problems

- DFS/BFS review, focus on state definition and DFS tree
- Bidirectional BFS
- Graph representation, BFS / DFS on graph
- taking care of any gap from DFS / BFS
- More data structure, stack, pair, etc.
- More coding skills and learning
- Introduction to more advanced topics, Disjoint Set Union, etc.

Class Structure

- All classes will be conducted on-line in zoom meetings.
- Students **must** join the google classroom as all class materials and announcements can be found there.
- All students **must** have their webcams turned on during the class
- All students are encouraged to proactively engaging in class activities such as:
  - answer questions
  - participate the breakout room discussions
  - participate in coding practice

Homework Policy

Each week a homework assignment will be provided to the students. They are expected to complete this **individually** and submit before the start of the next class. Discussion with friends/TA is encouraged.

If a student does not submit any homework for a 2 week period, we will contact the parent directly to see what is going on and how we can fix the issue. If it continues to 4 weeks, a meeting will be held to decide if it makes sense for the student to continue with the class.

Important!! Students should keep working on homework it if they didn't finish it in the previous week. Homework will be reviewed in next week.Students should finish the missing parts if there are any.

Homework

1

Review

Student spends time reviewing class materials to retain information

2

Self Study

Step 1:Pencil/paper

Step 2: Coding

Step 3: Debugging

3

Study Buddy

If stuck for more than 30 mins with no code, reach out to buddy for help

4

Office Hours

Get deeper insight and clear ideas from experienced TAs help

Exams and Reports

- Week 6 - Midterm Test
- Week 12 - Final Test
- Students' homework scores will be sent to their parents via email automatically every Wednesday, Friday and Saturday.
- Can also check using X-Wizard (instructions provided in handbook)
- Students' midterm and final reports will be sent to their parents in the week following the test
- Students who complete less than 60% of homework and score less than 60% on midterm/final may be required to withdraw
- 4.0 award

TA Office Hours

Office hours are provided to the students to get help with homework or ask questions about the material from class. They are **not mandatory**.

Students are encouraged to use these opportunities to get help for homework.

- Peter: Wednesday 7-8PM
- Alex: Thursday 7-8PM
- Cory: Friday 7-8PM

Zoom link will be the same used for class

THANK YOU

12280 Saratoga Sunnyvale Rd, STE 203, Saratoga, CA 95070

info@x-camp.org

http://x-camp.org

(650) 492-5695

Follow us on:

X-Camp Academy @XCampacademy Admin X-Camp x-camp-academy x-camp-academy

Let’s get to know each other

- What is your name?
- Which school/city are you from?
- Was you in X-Camp last quarter? What class?
- Anything else you want us to know about you?

About C++, if you were in Python class

- Review the materials in folder https://drive.google.com/drive/u/1/folders/12-ghH3auUmrUiWS6NVdgSHb4U56MUG-t
- Start using C++ for all coding problems
- Some practice problems: https://xjoi.net/contest/2601
- Google/internet is your friend E.g. https://www.cplusplus.com/

scanf/printf - faster alternatives of cin/cout

- When dealing with input/output, *cin/cout* are usually used
- While they are simple, they are slow
- Alternatively, we can use *scanf/printf*
- They are faster and can improve performance if you have a lot of input/output
- May be helpful to fix TLE (Time Limit Exceed) error

Scanf - faster alternative of cin

#include <cstdio>

int scanf (const char * format, ...);

Examples

```
printf ("Enter a number: ");
scanf ("%d", &n);// scanf requires the variable address
scanf ("%c",&ch); // char c
scanf ("%s %s", firstName, lastName); // char firstName[10]
```

Printf - faster alternative of cout

int printf (const char * format, ...);

Examples

```
    printf ("Characters: %c %c \n", 'a', 65);// endl
    printf ("Decimals: %d %ld\n", 1977, 650000L);
    printf ("Some different radices: %d %x %o %#x %#o \n",
  100, 100, 100, 100, 100);
    printf ("floats: %4.2f %+.0e %E \n", 3.1416, 3.1416, 3.1416);
  // output 3.14 for %4.2f
    printf ("%s \n", "A string");
```

Buddy Group

- Making a group with your classmate buddies
  - 3~4 people preferable
  - Discuss homework on discord or Zoom or other means
- When you have questions with class/homework
  - Check with your buddies or have group study with buddies
  - Go to TA office hour
- Benefits
  - Faster responses
  - Help each other, learn new opinions, enhance your understanding
  - Make new friends, promote friendship, etc.

X - Camp

review DFS

Floodfill & DFS & Maze Problems

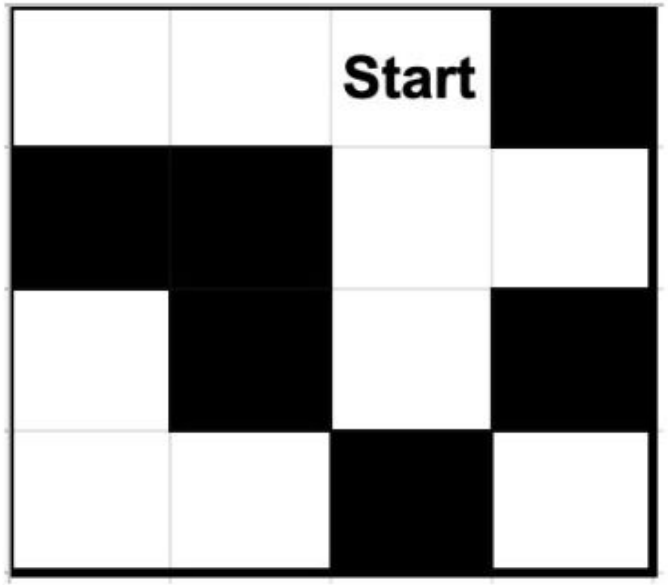
In this course, we will be more familiar with depth first search

Class : 202

Floodfill

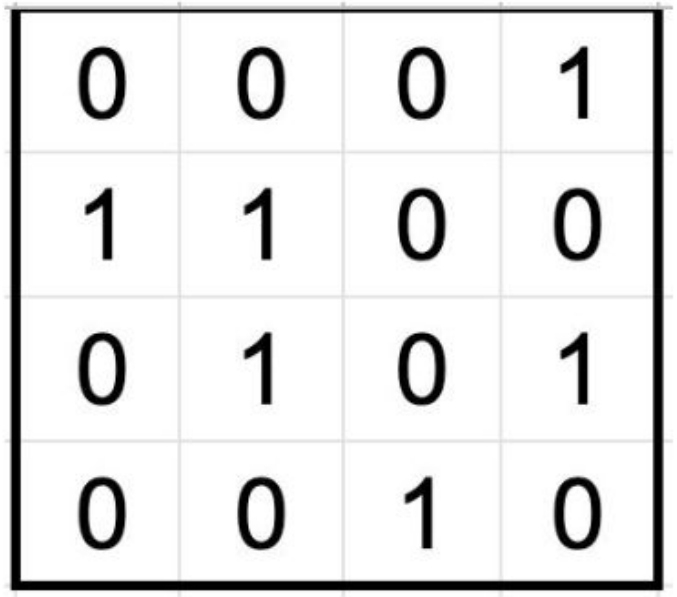
Connected block problem:
start from a starting point, count how many different points can be reached

(Each step you can move in 4 directions. - (Up, Down, Left, Right).)



0	0	0	1
1	1	0	0
0	1	0	1
0	0	1	0

Floodfill Step



Typical steps:

1. Take the position of the starting point.

2. Decide directions you want to go, e.g., (Up, Down, Left, Right).

3. Travel(flood) in those directions.

4. Repeat 2 and 3 till you've reached everywhere within the boundaries.

How to get to the next cell

```
void floodfill(int x, int y, ..) {
    // maintain answer related information
    …
    // Recursive calls
    floodfill(x + 1, y, ..);  // go south   (+1,   0)
    floodfill(x - 1, y, ..);  // or north   ( -1,   0)
    floodfill(x, y + 1, ..);  // or east    (0  , +1)
    floodfill(x, y - 1, ..);  // or west    (0  ,  -1)
}
```

0	0	0	1
1	1	0	0
0	1	0	1
0	0	1	0

What do you need?

We need :

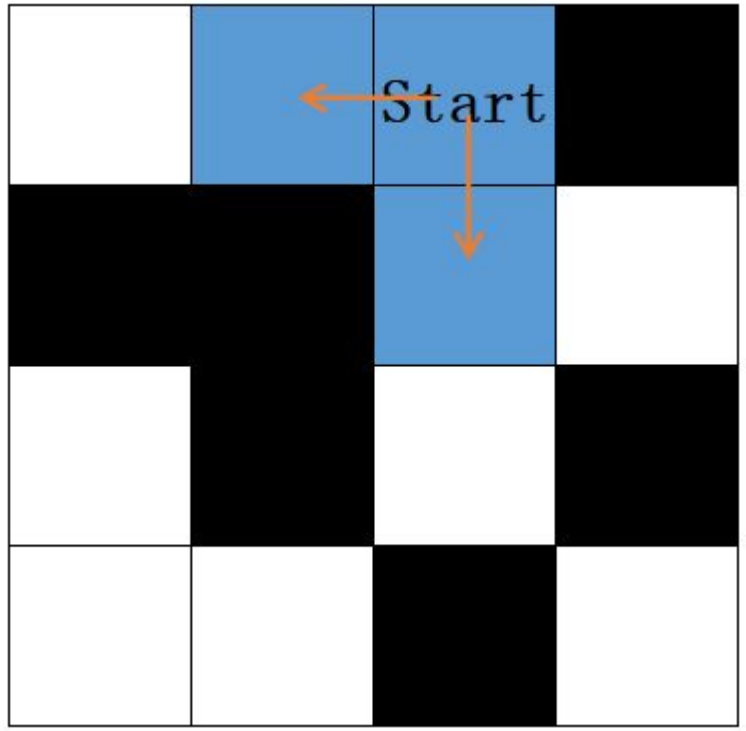
- A 2D array of maze :
  - **int / char maze[N][M]**
- A 2D array of marks:
  - **bool visited[N][M]**
- Count the cell reached:
  - A global counter variable: **int count**
- Two direction arrays :
  - **dirx = {0, 1, 0, -1}**
  - **diry = {1, 0, -1, 0}**

0	0	0	1
1	1	0	0
0	1	0	1
0	0	1	0

Determines where to go next

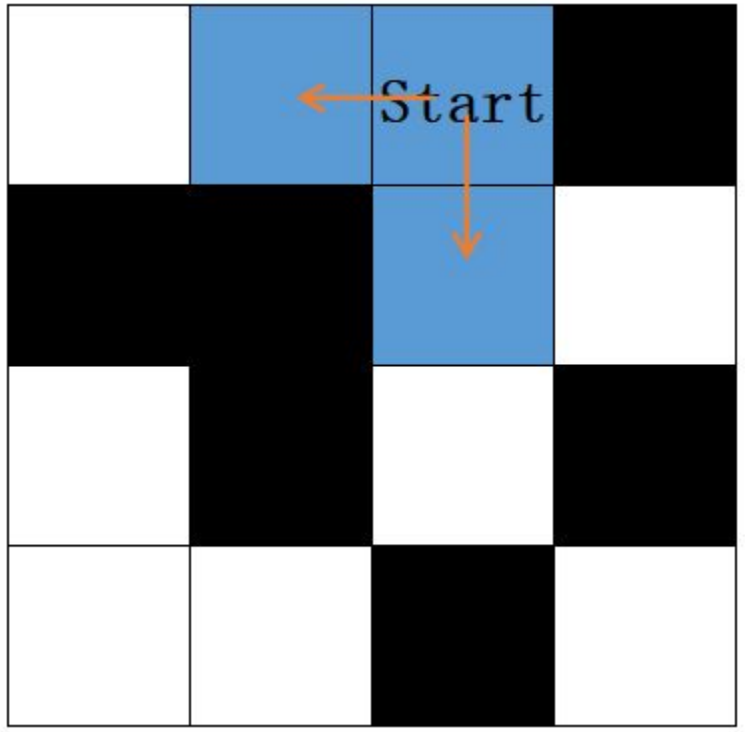
- Use a loop to try all directions to go:
  - **dirx = {0, 1, 0, -1}**
  - **diry = {1, 0, -1, 0}**

- Check if the next position is available
  1)In boundary? **(x >= 0 && x < N && y >= 0 && y < M)**
  2)Not visited yet? **(!vis[x][y])**
  3)Meet travel condition given by the problem
  - **maze[x][y] == 0 (normal maze)**
  - **maze[x][y] != maze[new_x][new_y] (0/1 maze)**



Sample code structure

```
void floodfill(int x, int y) {
      // mark as visited
      visited[x][y] = true;
      // maintain answer related information
            counter ++;
      // Recursive calls
      for (int i=0; i<4; i++) {
            new_x = x + dirx[i];
            new_y = y + diry[i];
            if ( isAvailable (new_x, new_y) )
                  floodfill (new_x, new_y);
      }
}
```



DFS - Floodfill (No Backtracking)

https://onlinegdb.com/Mys62-G2h

You are given a 2D maze. Find how many cells you can fill if you start from (0, 2)

Print out number of cells to be filled.

You should output 6.

0	0	0	1
1	1	0	0
0	1	0	1
0	0	1	0

If you have multiple queries (P8110)

There is a n×n maze which only has number 0 and 1. If you are in a spot 0, then you can move to one of the four adjacent cells with 1 in it. If you are in 1, then you can move to one of the four adjacent cells with 0 in it. Given the maze, find out how many cells you can move into (including the original cell).

Input:

The first line contains two positive integers n, m.

The next n lines, each line contains n characters (0 or 1, with no space in between)

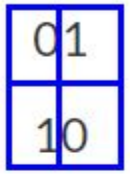
The next m lines, each line contains two positive integers i and j (separated by space), which means the cell in the i-th row and the j-th column (the starting point).

Output:

m lines, output the answer for each inquiry.

sample input 1:

2 2



1 1

2 2

sample output 1:

4

4

Note:

For 100% data, $n \le 1000, m \le 100000$。

Floodfill optimization for multiple inputs

If the number of connected blocks in the same area is queried multiple times

Calculate again? 

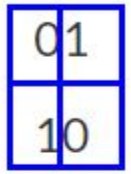
Save the results for calculated area.

```
//a global variable to save results
int results[area_id];
if (results[area_id] > 0) return results[area_id];
```

https://xjoi.net/contest/4362

sample input 1:

2 2



1 1

2 2

sample output 1:

4

4

Note:

For 100% data，$n \le 1000, m \le 100000$。

When to use floodfill?

- Area problem vs. path problem

- Area problem: to identify an reachable area from a start point
  - Just move on to reach all reachable spaces
  - Only need to visit each space once, no need for backtracking

- Path problem: to identify a path from start point to the end point
  - Need to do backtracking?

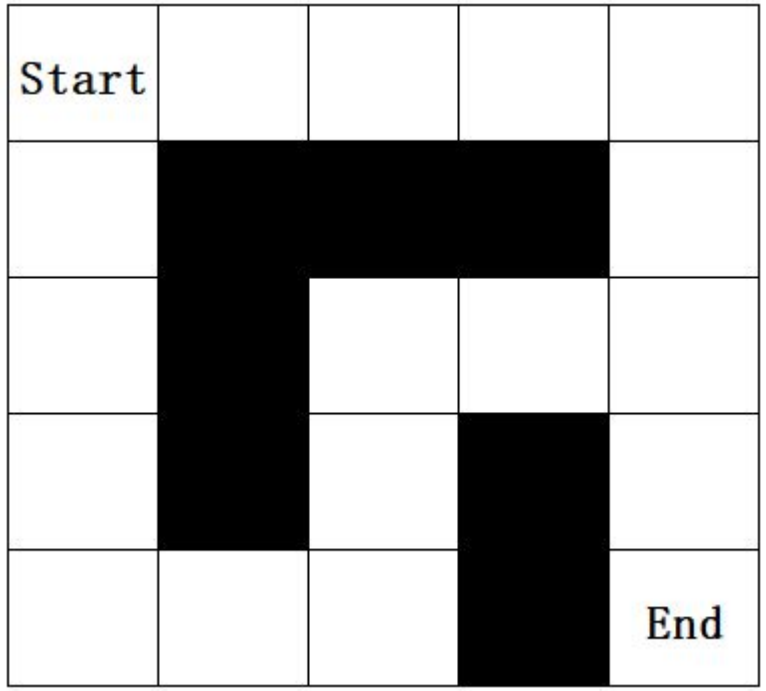
DFS - shortest path problem

Starting from the top left corner, how many steps are needed in minimum to reach the bottom right destination?

If there are multiple min paths, compute number of them as well.

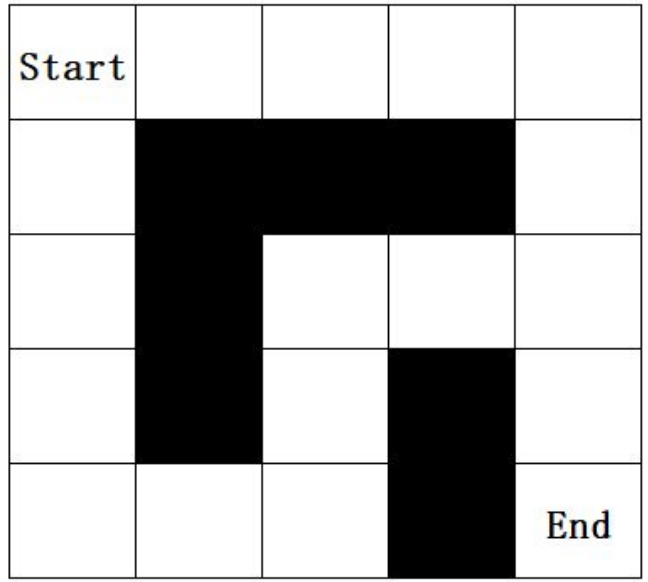
Note that there are some blocking cells.

Please take some time to code it.
https://onlinegdb.com/hjxfeL3ZS



DFS - Sample Recursion Codes

```cpp
int n,maze[100][100], ans=INT_MAX, ans_cnt=0; // Why ans is set to INT_MAX?
bool vis[100][100];
int dx[4] = {0, 1, 0, -1};
int dy[4] = {1, 0, -1, 0};
bool is_available(int x,int y) { // Check inside board && not visited && accessible
 return x>=0 && x<n && y>=0 && y<n && !vis[x][y] && maze[x][y] == 0;
}
void dfs(int x,int y,int steps) {
    if (x==n && y==n) {
        if (ans == steps) ans_cnt++;
        if (ans > steps) ans=steps, ans_cnt=1;
    }
    int tx,ty;
    for(int i=0;i<4;i++) { // Try four directions recursively
        tx=x+dx[i];
        ty=y+dy[i];
        if (is_available(tx,ty)) {
            vis[tx][ty]=1;
            dfs(tx,ty,steps+1);
            vis[tx][ty]=0; // Why? What happens if we remove this line?
        }
    }
}
```



Homework this week (P9561)

Friendship Circle

Given a n by n matrix M, representing the friendships in class. If M[i][j] = 1, i-th student and j-th student are friends to each other. Please compute the number of social circles in total.

Sample Input:

```
3

1 1 0

1 1 1

0 1 1
```

Sample Output:

```
 1
```

Homework this week (P8100)

Get water

> n people are lining in front of a tap to get some water. Suppose it takes Ti time for each person to get water. Please design a program to find out the sequence of those people in the line, so the average waiting time for those n people would be minimal.

Sample input 1 :

10

56 12 1 99 1000 234 33 55 99 812

Sample output 1 :

3 2 7 8 1 4 9 6 10 5

291.90

Homework this week (P9397)

Give a Row × Rol capital letter matrix, the starting position is the upper left corner. You can move up, down, left and right, and can't move to the letters that have passed. The question is how many letters you can go through at most.

[data format]
In the first row, enter the number of rows R and columns S of the alphabetic matrix (1 ≤ R, S ≤ 20). Then enter the letter matrix in row R and column S.

Output the maximum number of different letters that can be passed.

Sample input:

```
3 6
HFDFFB
AJHGDH
DGAGEH
```

Sample output:

```
6
```

Sample input:

```
3 6
HFDFFB
AJHGDH
DGAGEH
```