CS180 Discussion

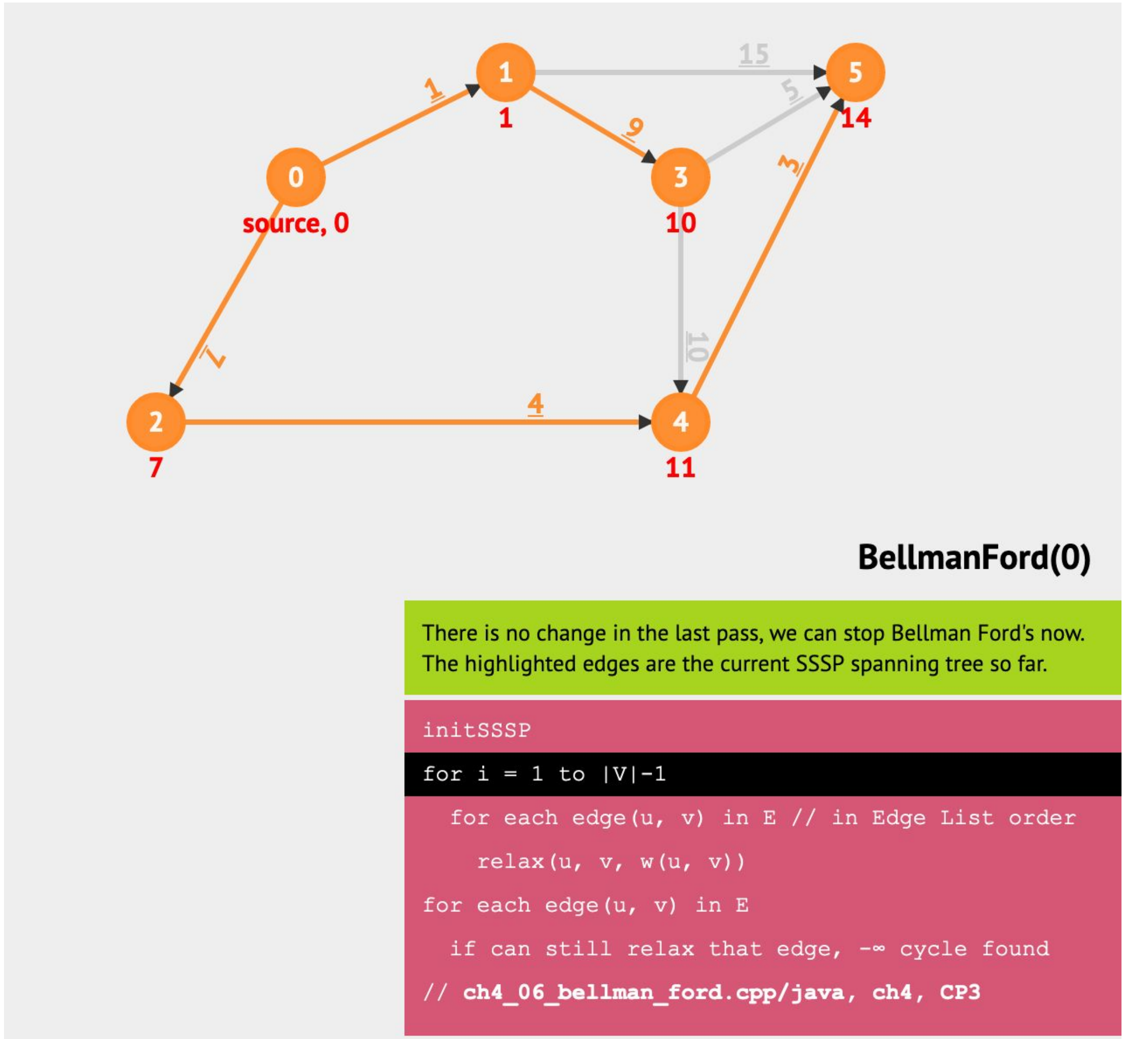
Week 6

Lecture Recap

- Bellman Ford
- Floyd Warshall
- Closest pair of points
- Celebrity problem

Bellman-Ford

Bellman-Ford vis

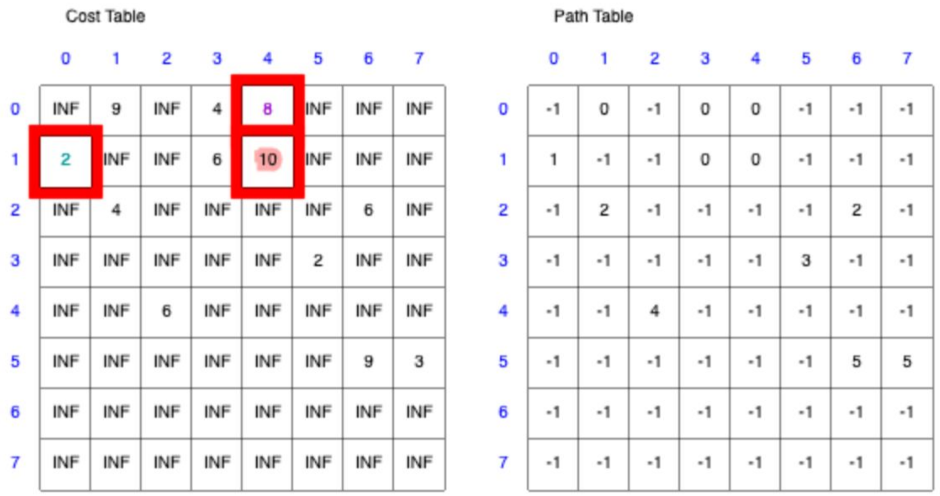


BellmanFord(0)

There is no change in the last pass, we can stop Bellman Ford's now.
The highlighted edges are the current SSSP spanning tree so far.

```
initSSSP
for i = 1 to |V|-1
  for each edge(u, v) in E // in Edge List order
    relax(u, v, w(u, v))
for each edge(u, v) in E
  if can still relax that edge, -∞ cycle found
// ch4_06_bellman_ford.cpp/java, ch4, CP3
```

Floyd-Warshall

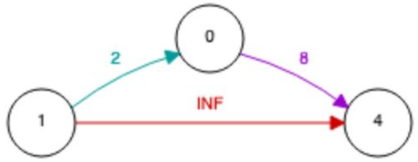


Cost Table

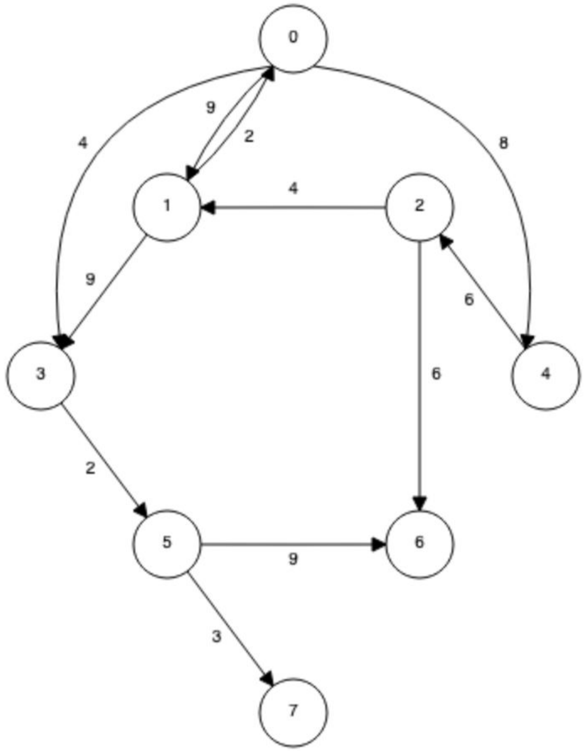
	0	1	2	3	4	5	6	7
0	INF	9	INF	4	8	INF	INF	INF
1	2	INF	INF	6	10	INF	INF	INF
2	INF	4	INF	INF	INF	INF	6	INF
3	INF	INF	INF	INF	INF	2	INF	INF
4	INF	INF	6	INF	INF	INF	INF	INF
5	INF	INF	INF	INF	INF	INF	9	3
6	INF	INF	INF	INF	INF	INF	INF	INF
7	INF	INF	INF	INF	INF	INF	INF	INF

Path Table

	0	1	2	3	4	5	6	7
0	-1	0	-1	0	0	-1	-1	-1
1	1	-1	-1	0	0	-1	-1	-1
2	-1	2	-1	-1	-1	-1	2	-1
3	-1	-1	-1	-1	-1	3	-1	-1
4	-1	-1	4	-1	-1	-1	-1	-1
5	-1	-1	-1	-1	-1	-1	5	5
6	-1	-1	-1	-1	-1	-1	-1	-1
7	-1	-1	-1	-1	-1	-1	-1	-1



2 + 8 < INF



Floyd-warshall vis

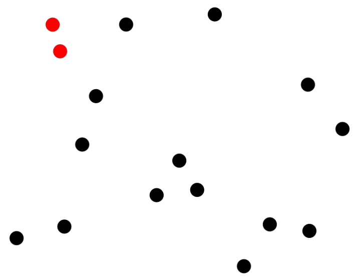
Closes Pair

We are given an array of n points in the plane. Find out the closest pair of points in the array.

The Brute force solution is O(n^2), compute the distance between each pair and return the smallest.

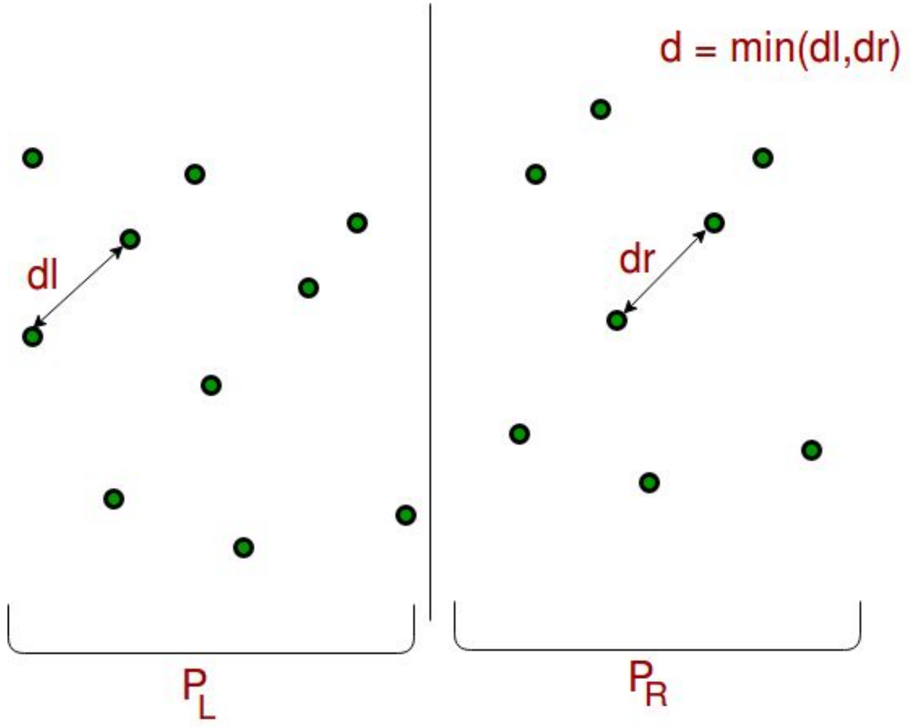
Closes Pair

1. Sort points according to their x-coordinates.
2. Split the set of points into two equal-sized subsets by a vertical line x=xmid.
3. Solve the problem recursively in the left and right subsets. This yields the left-side and right-side minimum distances d_{Lmin} and d_{Rmin}, respectively.
4. *Find the minimal distance d_{LRmin} among the set of pairs of points in which one point lies on the left of the dividing vertical and the other point lies to the right.
5. The final answer is the minimum among d_{Lmin}, d_{Rmin}, and d_{LRmin}.

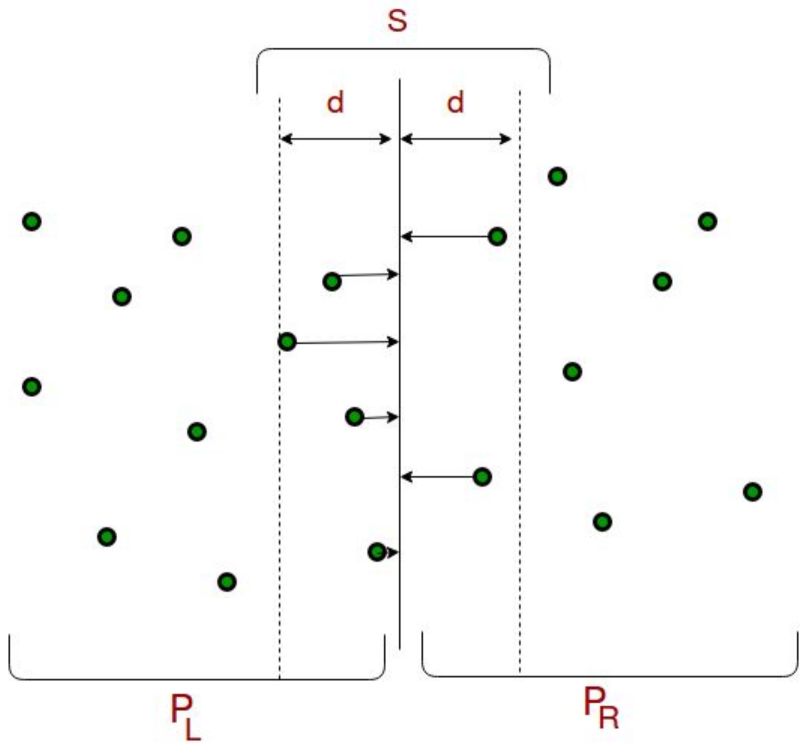


*We already know that the closest pair of points is no further apart than dist= min(d_{Lmin}, d_{Rmin}). Therefore, for each point *p* to the left of the dividing line we have to compare the distances to the points that lie in the rectangle of dimensions (dist, 2 · dist). And what is more, this rectangle can contain at most six points with pairwise distances at least d_{Rmin}.

Closes Pair



Step 3



Step 4

Celebrity Problem

A party of N people, only one person is known to everyone. Such a person may be present in the party, if yes, (s)he doesn’t know anyone in the party. We can only ask questions like “does A know B? “. Find the stranger (celebrity) in minimum number of questions.

Celebrity Problem Solution

- Ask 1 if they know 2:
  - If yes, we know that 1 is not famous and 2 could be famous
  - If no, we know that 1 could be famous and 2 is not famous
  - Eliminate one person with each step
- … for n - 1 questions
- n -> n - 1 -> … -> 2 -> 1 person in asking n - 1 questions, then go back and ask everyone if they know the last person and if the last person knows everyone else (additional 2(n - 1) questions)
- Improved complexity: 3(n - 1) ~ n
  - Went from n^2 to n

terview Questions!!!

Interview Questions!!!

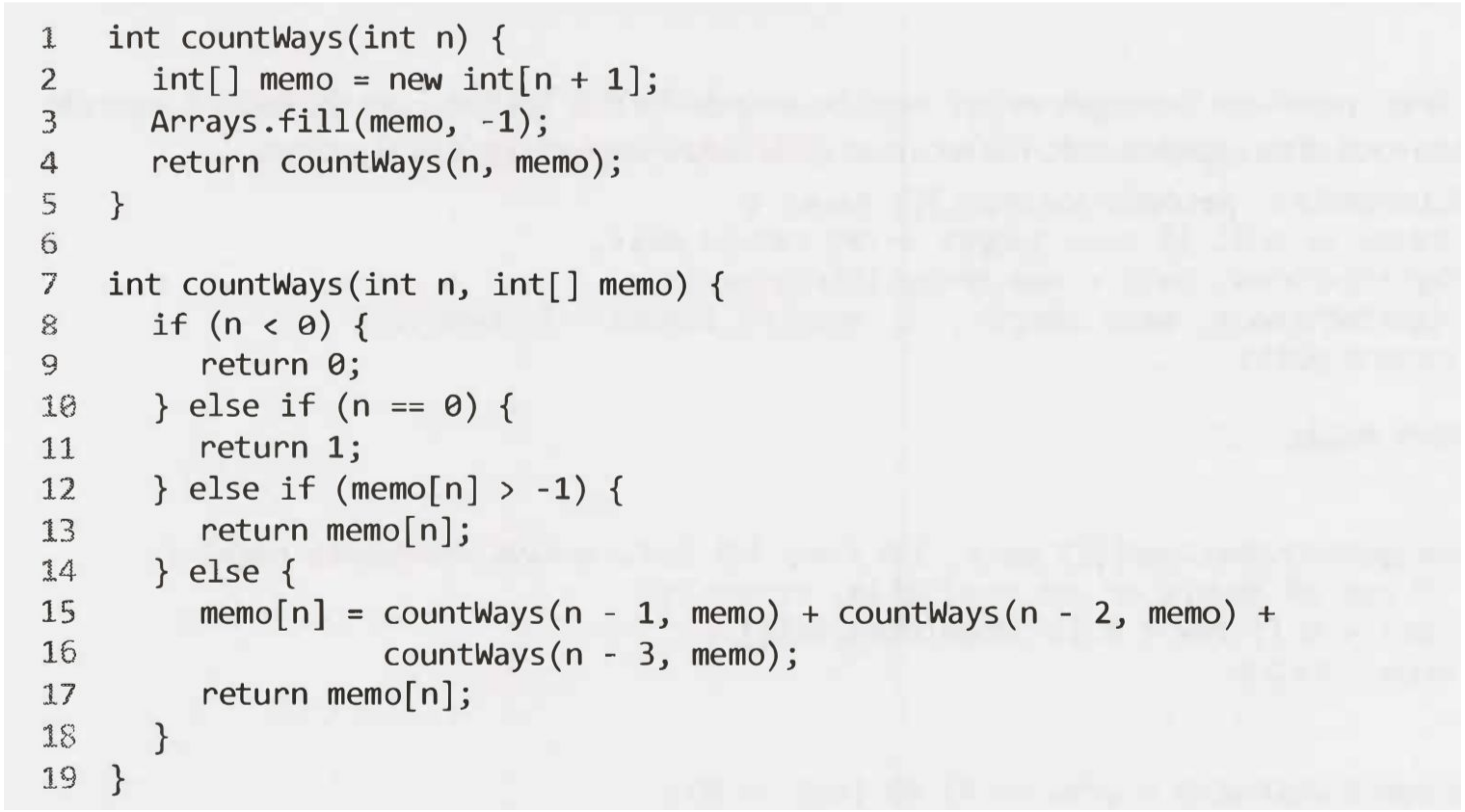
Interview Questions!!!

Interview Questions!!!

Triple Step

A child is running up a staircase with n steps and can hop either 1 step, 2 steps, or 3 steps at a time. Implement a method to count how many possible ways the child can run up the stairs.

Triple Step Code



```java
int countWays(int n) {
   int[] memo = new int[n + 1];
   Arrays.fill(memo, -1);
   return countWays(n, memo);
}

int countWays(int n, int[] memo) {
   if (n < 0) {
      return 0;
   } else if (n == 0) {
      return 1;
   } else if (memo[n] > -1) {
      return memo[n];
   } else {
      memo[n] = countWays(n - 1, memo) + countWays(n - 2, memo) +
                countWays(n - 3, memo);
      return memo[n];
   }
}
```