Dynamic Programming II

Coin Change?

How many ways to make **N** cents with pennies, nickels, dimes, and quarters?

Order matters

E.g. 6 = {(1,1,1,1,1,1), (1,5), (5,1)}

Coin Change?

What is the optimal subproblem?

- Make change for smaller **N**?
- Make change with fewer coins?

Coin Change?

```
ans[0] = 1
for (i = 1 to N)
  for(c in coins)
    if(i-c >= 0)
      ans[i] += ans[i-c];
```

Coin Change?

```
ans[0] = 1
for (i = 1 to N)
   for(c in coins)
      if(i-c >= 0)
         ans[i] += ans[i-c];
```

What is time complexity?

Coin Change II

How many ways to make **N** cents with pennies, nickels, dimes, and quarters?

Order **doesn't** matter

E.g. 6 = {(1,1,1,1,1,1), (1,5)}

Coin Change II

What is the optimal subproblem?

- Make change for smaller **N**?
- Make change with fewer coins?

Hmm…

Coin Change II

Simpler case: only pennies and nickels

Coin Change II

Simpler case: only pennies and nickels

Make 5**M** cents using only nickels, then
N-5**M** cents using only pennies

Coin Change II

What is the optimal subproblem?

- Make change for smaller **N** *and* with fewer coins
- ans[i,j] = ways to make **j** cents using first **i** coins

Coin Change II

simple test case: coins are {1¢, 2¢}

i \\ j	0	1	2	3	4	5
0						
1						
2						

ans[i,j] = ways to make **j** ¢ w/ first **i** coins

Coin Change II

simple test case: coins are {1¢, 2¢}

i \\ j	0	1	2	3	4	5
0	1	0	0	0	0	0
1						
2						

ans[i,j] = ways to make **j** ¢ w/ first **i** coins

Coin Change II

simple test case: coins are {1¢, 2¢}

i \\ j	**0**	**1**	**2**	**3**	**4**	**5**
0	1	0	0	0	0	0
1	1	1	1	1	1	1
2						

ans[i,j] = ways to make **j** ¢ w/ first **i** coins

Coin Change II

simple test case: coins are {1¢, 2¢}

i \\ j	0	1	2	3	4	5
0	1	0	0	0	0	0
1	1	1	1	1	1	1
2	1					

ans[i,j] = ways to make **j** ¢ w/ first **i** coins

Coin Change II

simple test case: coins are {1¢, 2¢}

i \\ j	**0**	**1**	**2**	**3**	**4**	**5**
0	1	0	0	0	0	0
1	1	1	1	1	1	1
2	1	1				

ans[i,j] = ways to make **j** ¢ w/ first **i** coins

Coin Change II

simple test case: coins are {1¢, 2¢}

i \\ j	**0**	**1**	**2**	**3**	**4**	**5**
0	1	0	0	0	0	0
1	**1**	1	**1**	1	1	1
2	1	1	2			

ans[i,j] = ways to make **j** ¢ w/ first **i** coins

Coin Change II

simple test case: coins are {1¢, 2¢}

i \\ j	**0**	**1**	**2**	**3**	**4**	**5**
0	1	0	0	0	0	0
1	1	**1**	1	**1**	1	1
2	1	1	2	2		

ans[i,j] = ways to make **j** ¢ w/ first **i** coins

Coin Change II

simple test case: coins are {1¢, 2¢}

i \\ j	0	1	2	3	4	5
0	1	0	0	0	0	0
1	**1**	1	**1**	1	**1**	1
2	1	1	2	2	3	

ans[i,j] = ways to make **j** ¢ w/ first **i** coins

Coin Change II

simple test case: coins are {1¢, 2¢}

i \\ j	**0**	**1**	**2**	**3**	**4**	**5**
0	1	0	0	0	0	0
1	1	1	1	1	**1**	1
2	1	1	**2**	2	3	

ans[i,j] = ways to make **j** ¢ w/ first **i** coins

Coin Change II

```
ans[0, 0] = 1
for (c = 0 to num_coins)
  for (i = 0 to N)
    if(c >= 1)
      ans[c, i] += ans[c-1, i]
    if(i – vals[c] >= 0)
      ans[c, i] += ans[c, i –
 vals[c]]
```

Coin Change II

```
ans[0, 0] = 1
for (c = 0 to num_coins)
  for (i = 0 to N)
    if(c >= 1)
      ans[c, i] += ans[c-1, i]
    if(i – vals[c] >= 0)
      ans[c, i] += ans[c, i – vals[c]]
```

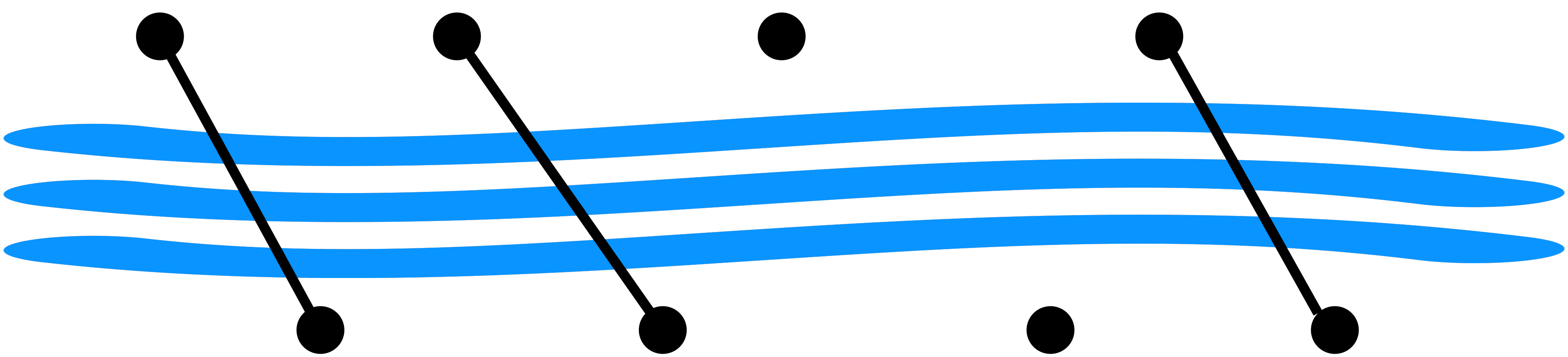
What is time complexity?

Bridge Building

Given points ($\mathbf{p}_i$,-1) and ($\mathbf{q}_i$,1) on opposite banks of river, build some bridges

Rules:

- no bridges can cross
- every ($\mathbf{p}_i$,-1) must have a bridge

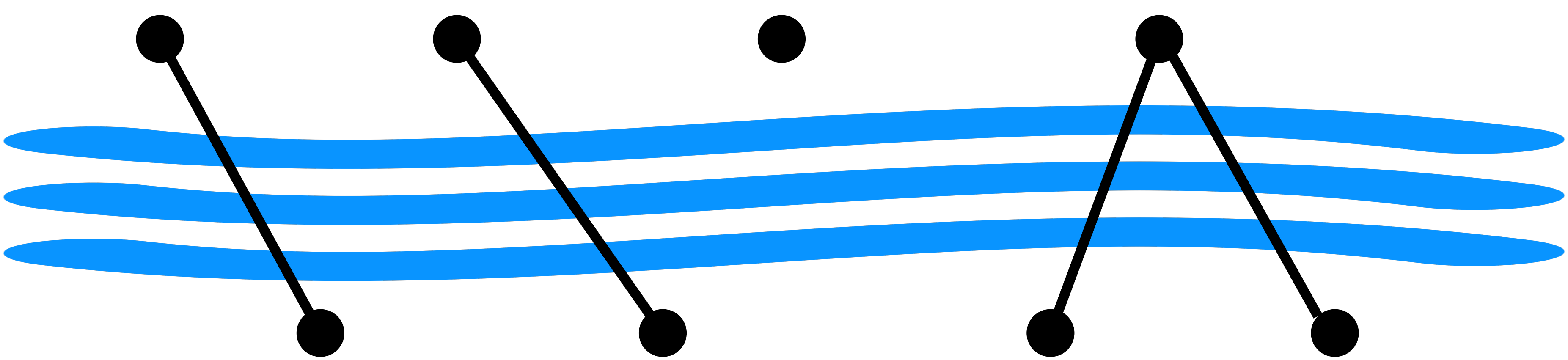


Bridge Building

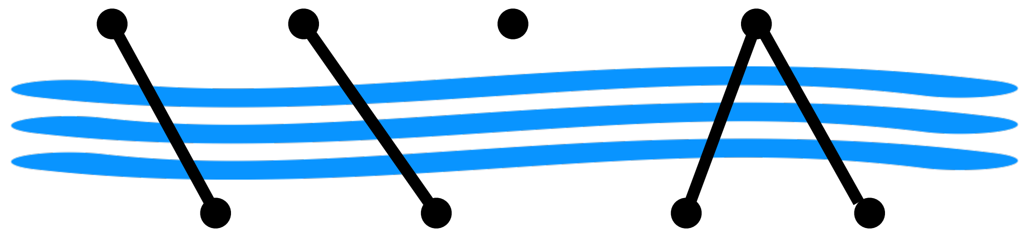
Given points ($\mathbf{p}_i$,-1) and ($\mathbf{q}_i$,1) on opposite banks of river, build some bridges

Rules:

- no bridges can cross
- every ($\mathbf{p}_i$,-1) must have a bridge



Bridge Building



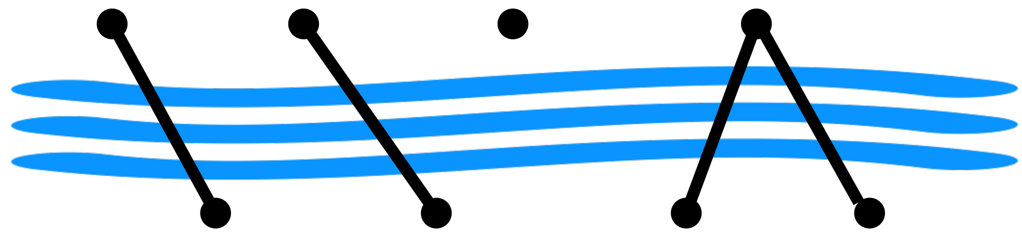
Given points ($\mathbf{p}_i$,-1) and ($\mathbf{q}_i$,1) on opposite banks of river, build some bridges

Rules:

- no bridges can cross
- every ($\mathbf{p}_i$,-1) must have a bridge

What is valid bridge network with shortest total length?

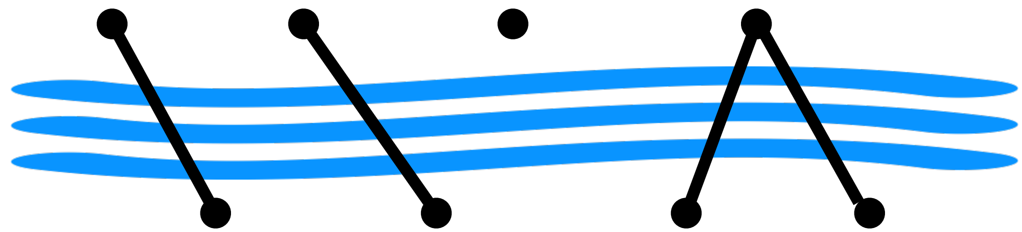
Bridge Building



Greedy algorithm

For each bottom point, connect it to nearest point on top

Bridge Building



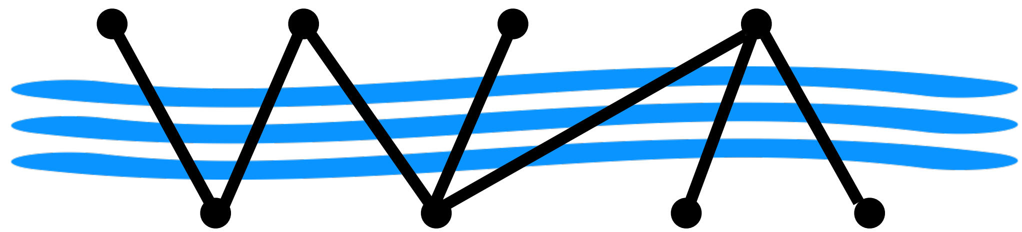
Given points ($\mathbf{p}_i$,-1) and ($\mathbf{q}_i$,1) on opposite banks of river, build some bridges

Rules:

- no bridges can cross
- every ($\mathbf{p}_i$,-1) must have a bridge
- must be **maximally-connected**

What is valid bridge network with shortest total length?

Bridge Building



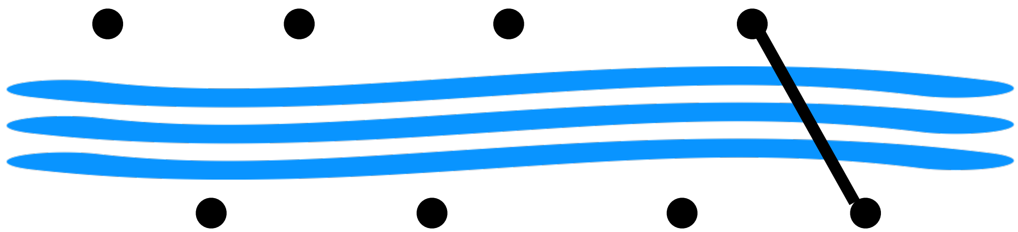
Given points ($\mathbf{p}_i$,-1) and ($\mathbf{q}_i$,1) on opposite banks of river, build some bridges

Rules:

- no bridges can cross
- every ($\mathbf{p}_i$,-1) must have a bridge
- must be **maximally-connected**

What is valid bridge network with shortest total length?

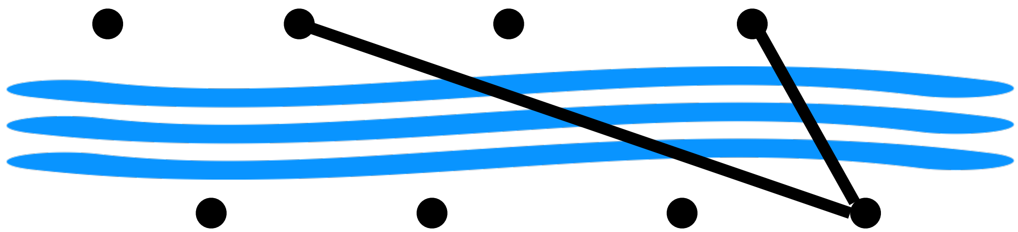
Bridge Building



Some observations:

- rightmost bottom point **must** connect rightmost top point

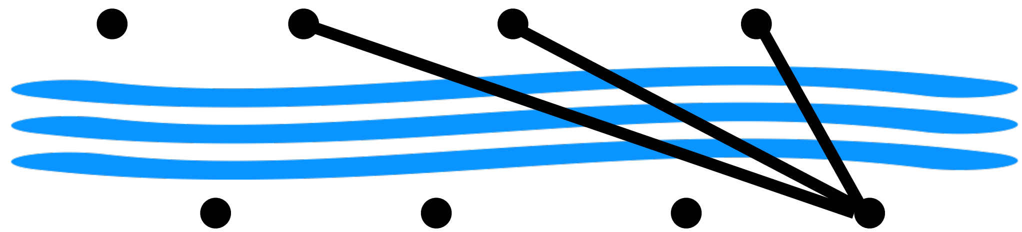
Bridge Building



Some observations:

- rightmost bottom point **must** connect rightmost top point
- could also connect to other top points

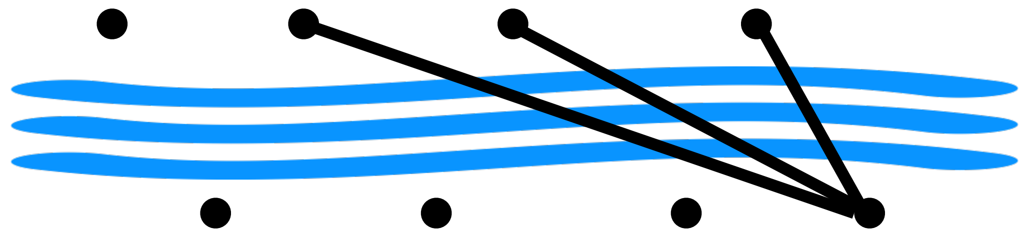
Bridge Building



Some observations:

- rightmost bottom point **must** connect rightmost top point
- could also connect to other top points
- if it connects to a top point, must connect to all other points to the right

Bridge Building



best(i,j) = shortest network using only up to bottom point **i** bottom and top point **j**

```
best(i,j):
  ans = infinity
  for (k = 0 to j)
    len = best(i-1,k)
    for (l = k to j)
      len += sqrt( (pi – ql)*(pi – ql) + 4 )
    ans = min(ans, len)
  return ans
```

Traveling Salesman

Given 16 cities and distances $\mathbf{d}_{ij}$ between them, what is cheapest itinerary that starts at city 0 and visits all cities exactly once?

Traveling Salesman

Given 16 cities and distances $\mathbf{d}_{ij}$ between them, what is cheapest itinerary that starts at city 0 and visits all cities exactly once?

Naïve solution: try 15! = ~1 trillion orders

Traveling Salesman

Given 16 cities and distances $\mathbf{d}_{ij}$ between them, what is cheapest itinerary that starts at city 0 and visits all cities exactly once?

Naïve solution: try 15! = ~1 trillion orders

What is the optimal subproblem?

Traveling Salesman

What is the optimal subproblem?

Find best paths visiting fewer cities?

Traveling Salesman

What is the optimal subproblem?

Find best paths visiting fewer cities?

- what if we **knew** that best path ended in city **c**?
- now can recurse

Traveling Salesman

best[**S**, **c**] = cheapest path visiting all cities in **S** and ending at city **c**

```
ans = infinity
for (x in S – {c, 0})
  len = best[S – {c}, x]
  len += d_xc
  ans = min(ans, len)
return ans
```

Traveling Salesman

best[**S**, **c**] = cheapest path visiting all cities in **S** and ending at city **c**

What is time complexity?

Traveling Salesman

best[**S**, **c**] = cheapest path visiting all cities in **S** and ending at city **c**

What is time complexity?

- 2^{15} subsets containing city 0
- 15 possible ending cities
- 2^{15}*15 ~= 500,000

Traveling Salesman

best[**S**, **c**] = cheapest path visiting all cities in **S** and ending at city **c**

What is time complexity?

- 2^{15} subsets containing city 0
- 15 possible ending cities
- 2^{15}*15 ~= 500,000
- NP-hard problem… no free lunch