Dynamic Programming

We want to find the choices that maximize or minimize some quantity.

Rod Cutting Example
We are given a rod of length n and a table of prices p_i for i = 1, . . . , n; p_i is the price of a rod of length i.
Goal is to determine the maximum revenue r_n, obtainable by cutting up the rod and selling the pieces

Try to solve this for n=5 and $p_1=1, p_2=5, p_3=8, p_4=9, p_5=10$

Dynamic Programming

We want to find the choices that maximize or minimize some quantity.

Rod Cutting Example
We are given a rod of length n and a table of prices p_i for $i = 1, \ldots, n$; p_i is the price of a rod of length i.
Goal is to determine the maximum revenue r_n, obtainable by cutting up the rod and selling the pieces

Notice that if the optimal solution has a cut of length i, and we took away that length i, what remains must be an optimal solution for a rod of length $n' = n-i$. Why?

Dynamic Programming

We want to find the choices that maximize or minimize some quantity.

Rod Cutting Example:
We are given a rod of length n and a table of prices p_i for i = 1, . . . , n;
Goal is to determine the maximum revenue r_n, obtainable by cutting up the rod and selling the pieces

We can recursively define a solution by figuring out where to make the first cut to maximize our possibilities with what's leftover.
We want the maximum of
$p_1 + R(n-1), p_2 + R(n-2), p_3 + R(n-3), \ldots, p_{n-1} + R(1), p_n$

Write a recursive algorithm R(n, p[1..n]) which does this

Dynamic Programming

We want to find the choices that maximize or minimize some quantity.

```
R(n, p[1..n]){
 if (n==1): return p[1]
 else{   max = 0;
         for i = 1 to n{
          x= p[i]+R(n-i)
               if(x>max): max=x
         }
        return x
}
```

If we called R(4), how many times each would R(1), R(2), and R(3) be called?

Bottom-up Dynamic Programming

Don't wait until until a subproblem is encountered. Solve smallest subproblems first and combine solutions of small subproblems to solve larger ones.

```
R(n, p[1..n]){
 r[0] = 0
 for (j=1; j<=n; j++){
     max = 0;
     for (i=1; i<=j; i++){
             x= p[i]+r[j-i]
            if(x>max): max=x
      }
      r[j] = max
  }
  return r[n] }
```

Iteratively fill in r array by calculating r[1],r[2],r[3] . . .for n=5 and $p_1=1, p_2=5, p_3=8, p_4=9, p_5=10$

Bottom-up Dynamic Programming

Don’t wait until until a subproblem is encountered. Solve smallest subproblems first and combine solutions of small subproblems to solve larger ones.

```
R(n, p[1..n]){
 r[0] = 0
 for (j=1; j<=n; j++){
     max = 0;
     for (i=1; i<=j; i++){
              x= p[i]+r[j-i]
            if(x>max): max=x
      }
      r[j] = max
  }
  return r[n] }
```

What is the order of growth?

Bottom-up Dynamic Programming

Don't wait until until a subproblem is encountered. Solve smallest subproblems first and combine solutions of small subproblems to solve larger ones.

```
R(n, p[1..n]){
 r[0] = 0
 for (j=1; j<=n; j++){
     max = 0;
     for (i=1; i<=j; i++){
             x= p[i]+r[j-i]
           if(x>max): max=x
      }
      r[j] = max
  }
  return r[n] }
```

Modify the code so it outputs the cuts rather than just the max profit.

Dynamic Programming Practice

Problem: Given an array A[1..n] of real numbers, find the numbers j and k so that the sum A[j..k] is maximal

Step1: Let S[k] represent the maximal subsequence that ends in position k. Write a recursive definition for S[k+1]

Dynamic Programming Practice

Problem: Given an array A[1..n] of real numbers, find the numbers j and k so that the sum A[j..k] is maximal

$S[0] = 0$

$S[k+1] = \max\{S[k] + A[k+1], A[k+1]\}$

Step2: Write bottom-up code to compute S[k] and use another array T[k] to store the starting index

Dynamic Programming Practice

Problem: Given a graph with vertices labeled 1 … n, find the shortest path between all possible pairs of vertices.

Step1: Let D[i,j,m] represent the shortest path cost from vertex i to vertex j with at most m edges
Let w[i,j] be the weight of edge from vertex i to vertex j.
Write a recursive definition for D[i,j,m+1]

Dynamic Programming Practice

Problem: Given a graph with vertices labeled 1 … n, find the shortest path between all possible pairs of vertices.

D[i,j,0] = 0 if i=j and infinity otherwise

D[i,j,m+1] = min{D[i,j,m], min 0<k<=n {D[i,k,m] + w[k,j]}}

Step2: Write bottom up code to compute D[i,j,n] for all pairs of i and j

What is the order of growth?

Dynamic Programming Practice

Problem: Given a graph with vertices labeled 1 … n, find the shortest path between all possible pairs of vertices.

Step1: Let D[i,j,m] represent the shortest path cost from vertex i to vertex j **which only uses paths through vertices 1 through m.**
Let w[i,j] be the weight of edge from vertex i to vertex j.
Write a recursive definition for D[i,j,m+1]

Dynamic Programming Practice

Problem: Given a graph with vertices labeled 1 … n, find the shortest path between all possible pairs of vertices.

$D[i,j,0] = w[i,j]$

$D[i,j,m+1] = \min\{D[i,j,m], D[i,m+1,m] + D[m+1,j,m]\}$

Step2: Write bottom up code to compute $D[i,j,n]$ for all pairs of i and j

What is the order of growth?