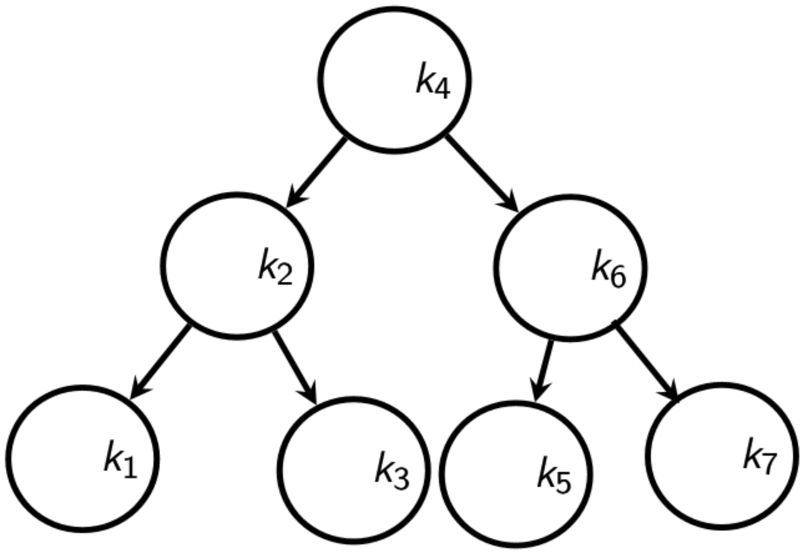
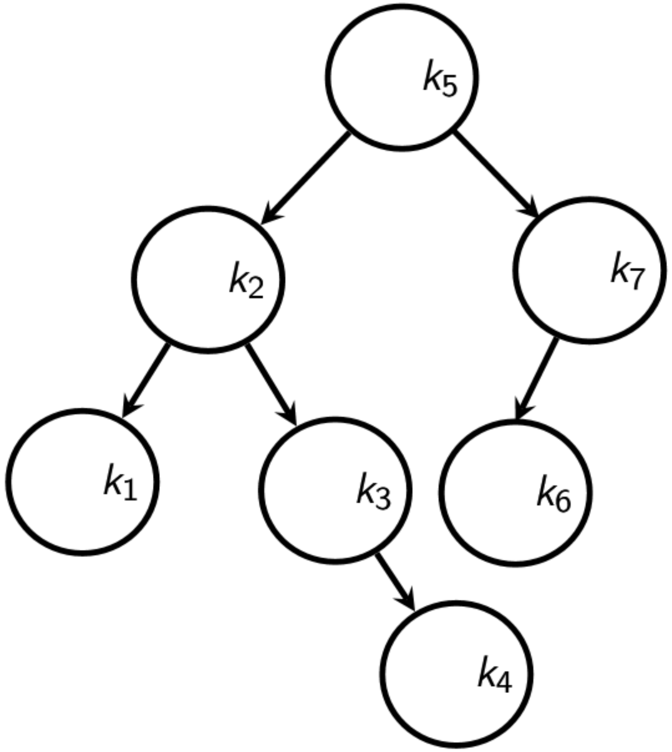
Please first refer to the handout from yesterday.

CSCI 570 Summer 2016
Dynamic Programming III

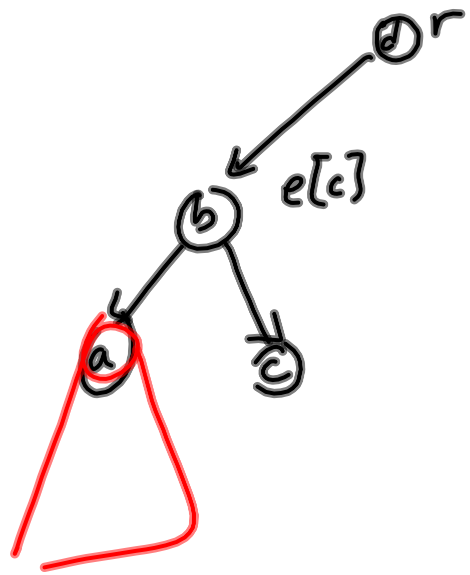
First Tree



Second Tree



Expected lookup cost: $\sum_{i \in T} d_i \cdot p_i$



totprob[a,c]

$p_d \cdot 1 + (p_a + p_b + p_c) + e[c]$

totprob[i,j]

OPT(i..j)

+ totprob[i..r-1]

totprob[r+1..j]

+ p[r]

Recursive Solution

OPT(i,j): optimal cost of BST on nodes i..j (inclusive)

if i==j return p[i]

if j<i return 0

only happens if i=j+1

// if r = optimal root of i...j

// I would return

$\min_{i \le r \le j} \{ OPT[i,r-1] + OPT[r+1,j] + \text{totprob}[i,j]$

running time? each element? O(n) how many? O(n²) total: O(n³)

precompute: O(n²)

Iterative Solution

Can't: for i=1 to n
for j=1 to n
fill in OPT[i,j]

```
for i=1 to n-1
    OPT[i,i]=Pi
    OPT[i+1,i]=0
OPT[n,n]=Pn
for δ=1 to n-1          // length δ subproblem?
    for i=1 to n-δ      // (how many nodes?)
        j=i+δ ; tot=totprob[i,j]
        // now fill in OPT[i,j]
        Min= OPT[i,i-1] + OPT[i+1,j] + tot
        bestr= i
        for r=i+1 to j
        {   cost = OPT[i,r-1]+ OPT[r+1,j] + tot
            if cost<Min
                Min=cost
                bestr=r
        }
        OPT[i,j]=Min
        best[i,j]=bestr
```

Produce the tree

return root of optimal BST on i..j (w/ children attached)

```
Node * produce(i,j)
{
    if i>j return nullptr;
    r = bestr[i,j]
    Node * n = new Node;
    n -> key = r;
    n -> left = produce(i,r-1)
    n -> right = produce(r+1,j)
    return n;
}
```

Example

$B \cdot C \cdot D$

$(B \cdot C) \cdot D$

$B \cdot C : 1000$

$(B \cdot C) \cdot D : 2000$

3000

$B \cdot (C \cdot D)$

$C \cdot D : 10{,}000$

- B is 2×10
- C is 10×50
- D is 50×20

$d = \{2, 10, 50, 20\}$

M_i is $d_{i-1} \times d_i$

Recursive Solution

$OPT(i,j)$: optimal # scalar mult to produce $M_i \cdots M_j$

if $i=j$ return 0 // matrix already exists

if $j=i+1$ return $d_{i-1} \times d_i \times d_{i+1}$

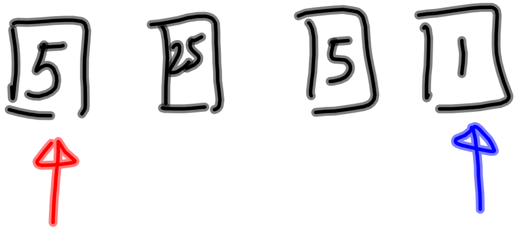
// must split like $(M_i \ldots M_k)(M_{k+1} \ldots M_j)$

return $\min\limits_{i \le k < j} \left\{ OPT(i,k) + OPT(k+1,j) + d_{i-1} \times d_k \times d_j \right\}$

Iterative Solution

(just like optimal binary search trees)

Why not greedy?



Recursive Solution

OPT(i,j): most pts obtainable, cards i..j on table

if $j < i$ return 0 //no cards

if $j = i$ return V_i

if $j = i+1$ return $\max(v_i, v_j)$

cardi = $V_i + \min(OPT(i+2, j), OPT(i+1, j-1))$ // opp. has $i+1 \ldots j$ // can give me $i+2 .. j$ or $i+1 .. j-1$

cardj = $V_j + \min(OPT(i+1, j-1), OPT(i, j-2))$

return Max(cardi, cardj)

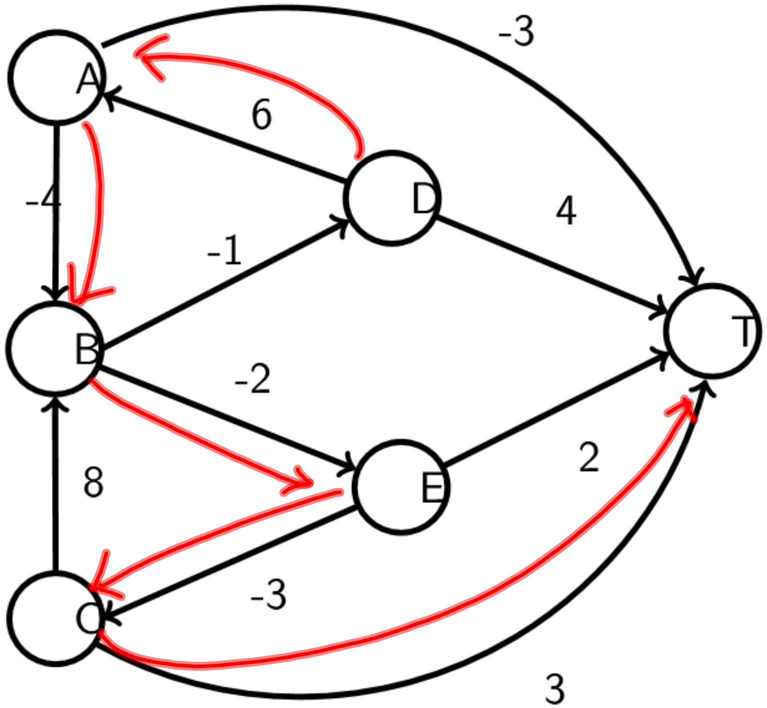
- What is the longest, in terms of the number of edges, that a shortest path could be? $n-1$
- What would it mean if a shorter path had more edges than that? Must be neg cost cycle

all positive: use Dijkstra's

What are the shortest paths to T?

plural

single dest shortest path



Recursive Solution

OPT(i,v) //shortest $v \rightsquigarrow T$ with $\le i$ edges

if v==T return 0
elif i=0 return ∞
else
stay = OPT(i-1,v)
go = $\min_{w \in adj[v]} \{c(v,w) + OPT(i-1,w)\}$
return Min(stay,go)

Iterative Solution

```
declare OPT[0...n-1, 1...|V|]
for i = 0 to n-1    OPT[i,T] = 0
for each v ∈ V-{T},  OPT[0,v] = ∞
declare next[1...|V|]
for i = 1 to n-1
    for each v ∈ V
        best = OPT[i-1,v]
        for each w ∈ adj[v]
            cost = OPT[i-1,w] + c[v,w]
            if cost < best
                best = cost
                next[v] = w
        OPT[i,v] = best
```

Illustration

$O(n^2)$ memory

	0	1	2	3	4	5
T	0	0	0	0	0	0
A	∞	-3	-3	-4		
B	∞	∞	0	-2		
C	∞	3	3	3		
D	∞	4	3	3		
E	∞	2	0	0		

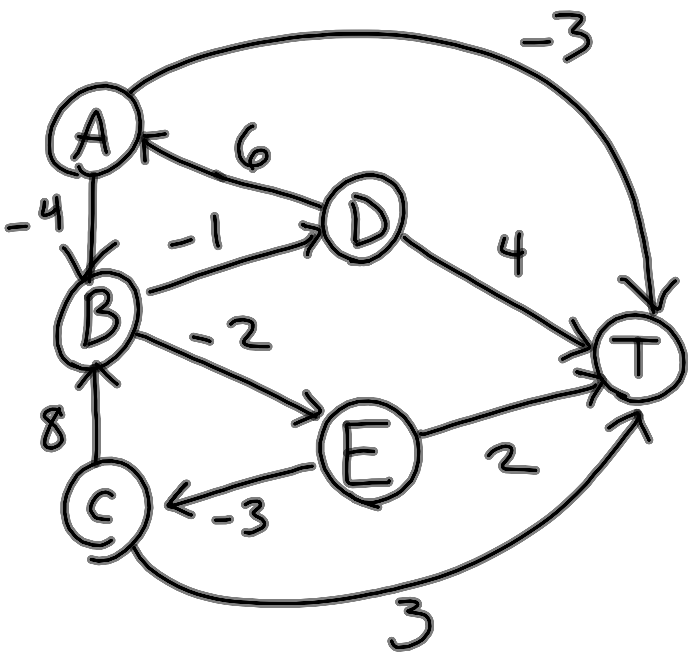
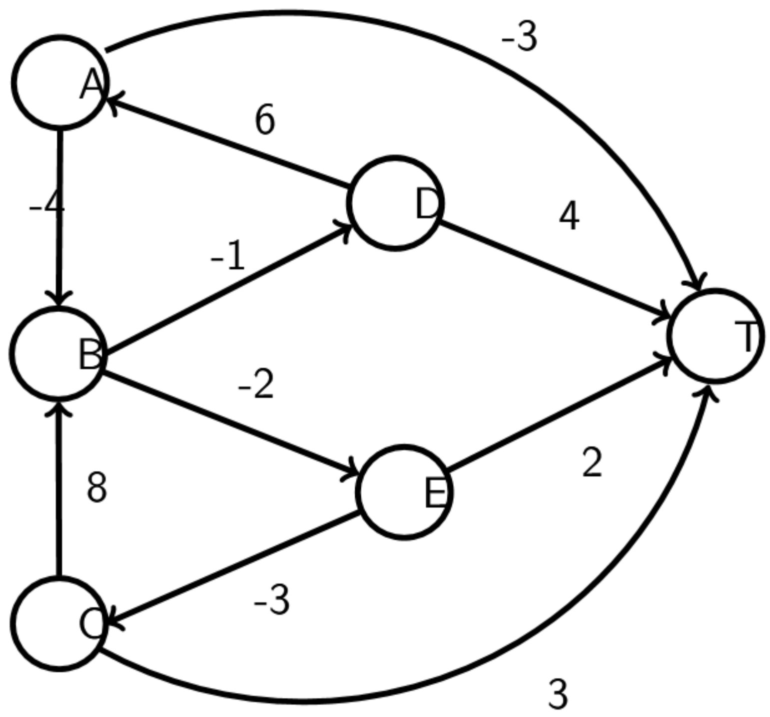


Illustration (2)



Can we use less memory?

- Keep 2 cols: current & previous

or even one:

$$M[v] \approxeq OPT(?, v)$$

$$= \min\left(M[v], \min_{w \in adj[v]} \{ M[w] + c[v,w] \} \right)$$

base...

Iter: for $i=1$ to n

$M[v] = $ - —

Can you produce the shortest path tree?

- see amended alg (w/ "next" ptrs)
- Tree: Vertices same

  Edges: all (v, next[v]) pairs

Can we detect negative-cost cycle(s)?

- +1 iter, if change → neg cost cycle
- or "next ptr tree" has a cycle at end