CSE 201 2-13-20

---

Discrete
Continuous ... Knapsack Problem

$$x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$$

is called a feasible solution

iff

$$\sum_{i=1}^{n} x_i w_i \leq W$$

Problem: amongst all feasible solutions, find one s.t.

$$\underbrace{\sum_{i=1}^{n} x_i v_i}_{\uparrow} \text{ is maximum}$$

objective function.

Define:

$V[i,j]$ = maximum value of objects in $\{1, 2, \ldots, i\}$ whose total weight does not exceed $j$

where $1 \le i \le n$, $0 \le j \le W$

note: when $i = 0$, $\{1, 2, \ldots, i\} = \emptyset$.

Really 2 Problems:

(1) find value of OPT. solution
i.e. $V[n, W]$

(2) find an optimal set of objects to steal: $x = (x_1, x_2, \ldots, x_n)$

Two alternatives

- Do not include obj $i$. In that case

$$V[i,j] = V[i-1,j]$$

- Do include obj $i$. then

$$V[i,j] = v_i + V[i-1, j-w_i]$$

In general

$$V[i,j] = \max\left(V[i-1,j],\ v_i + V[i-1,\ j-w_i]\right)$$

Include out-of-bounds values

$$V[i,j] = \begin{cases} 0 & (i=0 \text{ & } j \geq 0) \\ \max(V[i-1,j],\ v_i + V[i-1, j-w_i]) & (i>0 \text{ & } j \geq 0) \\ -\infty & (j<0) \end{cases}$$

Ex. $n = 5$

$w = (1, 3, 5, 6, 7)$

$v = (1, 5, 12, 25, 30)$

$W = 10$

| i | $w_i$ | $v_i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 3 | 5 | 0 | 1 | 1 | 5 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| 3 | 5 | 12 | 0 | 1 | 1 | 5 | 6 | 12 | 13 | 13 | 17 | 18 | 18 |
| 4 | 6 | 25 | 0 | 1 | 1 | 5 | 6 | 12 | 25 | 26 | 26 | 30 | 31 |
| 5 | 7 | 30 | 0 | 1 | 1 | 5 | 6 | 12 | 25 | 30 | 31 | 31 | 35 |

i

Exercise

Write pseudo-code for algorithm that fills table, given $V[1 \ldots n]$, $w[1 \ldots n]$, $W$.

Exercise

Write a recursive algorithm that, given filled table, backtracks through table, prints optimal seq. of objects to steal.

# Principle of Optimality (optimal sub-structure)

Any solution to a non-trivial instance is a combination of optimal solutions to some of its sub-instances.

- coin change

$$C[i,j] = \min\left(C[i-1,j],\ 1 + C(i, j-d_i)\right]$$

- Knapsack

$$V[i,j] = \max\left(V[i-1,j],\ v_i + V[i-1, j-w_i]\right)$$

Defn

Given graph $G$, and $u, v \in V(G)$

- a $u$-$v$ Path is a seq. of distinct vertices, any consecutive Pair of which are adjacent

$$u = x_0, x_1, x_2, \ldots, x_k = v$$

- length of Path is $k$, # of edge traversals
- $d(u,v)$ = length of a min length $u$-$v$ Path
- $\ell(u,v)$ = length of a longest $u$-$v$ Path.

Problem 1

Given $G$, and $u, v \in V(G)$ find $d(u,v)$ and a path of length $d(u,v)$

Problem 2

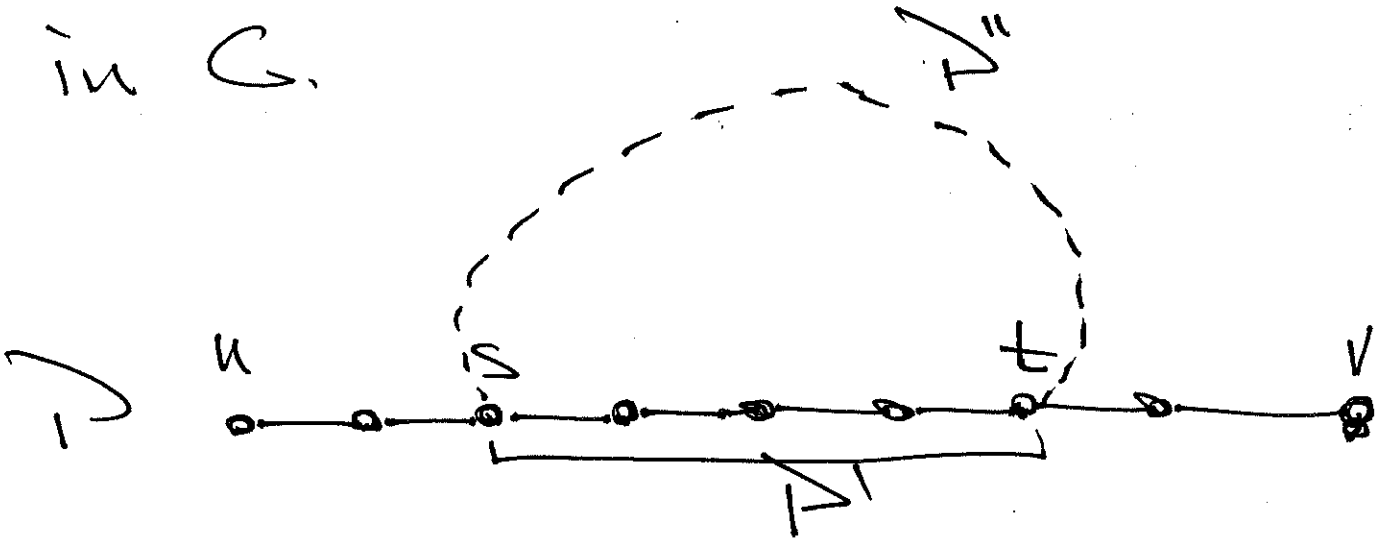
..... find $\ell(u,v)$ and a path of length $\ell(u,v)$

Theorem

Any subsequence of a shortest Path is also a shortest path

Proof

Let $p$ be a shortest u-v Path in G.




Let $s, t \in V(G)$ intermediate along $p$. Let $p'$ be the subsequence consisting of $p$ from s to t. We must show $p'$ is a shortest s-t Path.

Assume, to get a $\not\downarrow$, that $P'$ is not a shortest s-t Path. Then, there exists some shorter s-t path, Call it $P''$ .....

let

$P_{us}$ : $P$ from u to s

$P_{tv}$ : $P$ from t to v

then

$$\text{length}(P_{us}, P'', P_{tv})$$

$$= \text{len}(P_{us}) + \text{len}(P'') + \text{len}(P_{tv})$$

$$< \text{len}(P_{us}) + \text{len}(P') + \text{len}(P_{tv}).$$

$$= \text{length}(P_{us}, P', P_{tv})$$

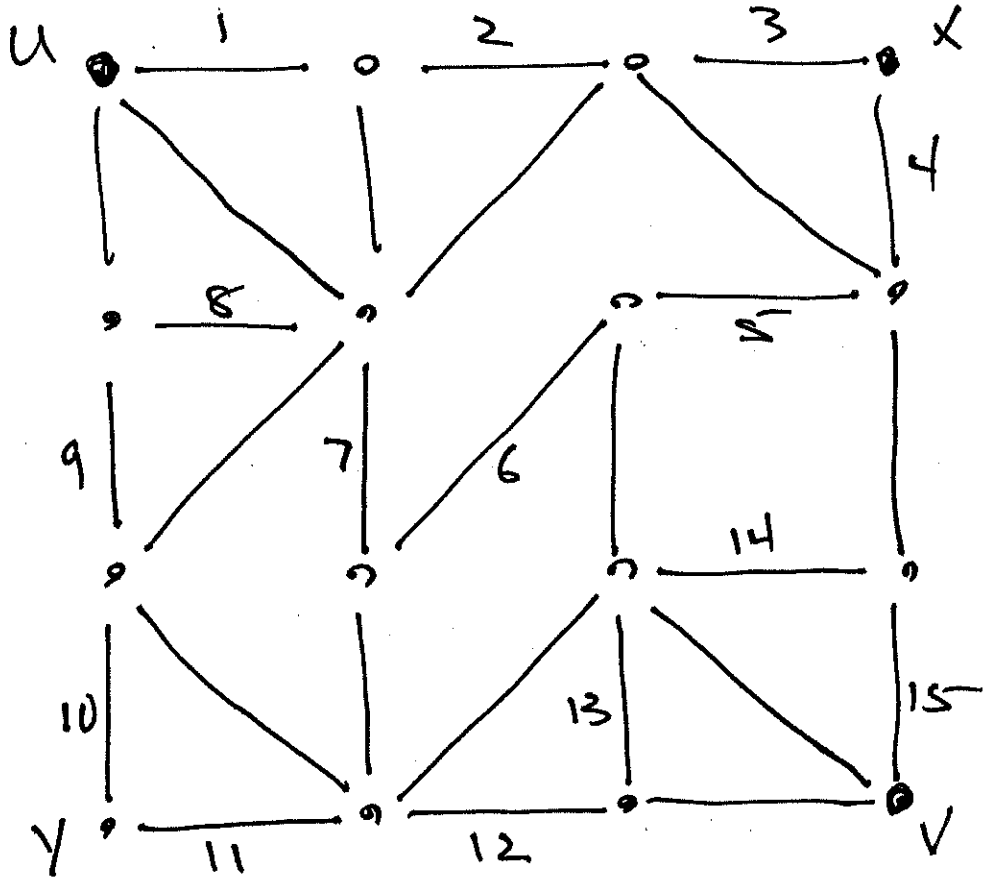
$$= \text{length}(P)$$

Thus $P_{us}, P'', P_{tv}$ (concatenated) is a u-v path shorter than $P$, impossible.

Therefore no such $P''$ can exist ∎

what about longest Path ?

Ex.




$\ell(u,v) = 15$

Subseq. from x to y is not a longest x, y Path, uses only 7 edges.

check: $\ell(x,y) \geq 14$.

Thus longest path does not exhibit optimal substructure.

Procedure for designing Dyn. Prog. Algorithms:

1. characterize the structure of an optimal solution as a combination of optimal subinstance solutions
2. Recursively define value of opt. solution in terms of values of opt. sub instance solutions.

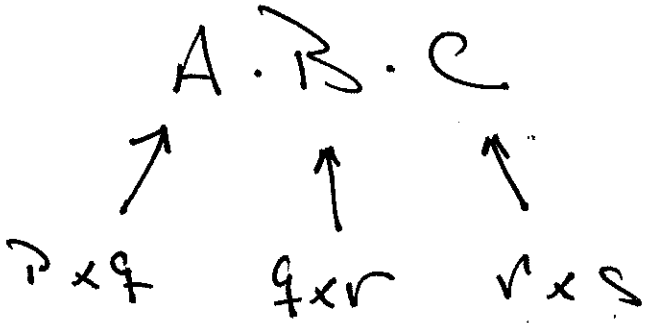
3. Compute value of an opt. solution in a bottom-up fashion, i.e. fill in the table.

4. Construct an optimal solution by back-tracking through table.

## Problem

Matrix-chain multiplication

Ex. $A \cdot B \cdot C$




cost = # of real no. multiplications

$$(A \cdot B)_{ij} = \sum_{k=1}^{q} a_{ik} b_{kj}$$ { q multiplications

do for $1 \le i \le p$, $1 \le j \le r$

total cost = $pqr$

$(A \cdot B) \cdot C$ : cost = $pqr + prs$

$A \cdot (B \cdot C)$ : cost = $pqs + qrs$