

If least-cost paths are desired:

For all vertices i, j let $X[i, j]$ store an intermediate vertex u appearing on the least-cost path from i to j where $u \neq i$, and $u \neq j$

for $i := 1$ to n do

for $j := 1$ to n do

$X[i, j] := 0$

if $(i, j) \in E$

then $C[i, j] := cost(i, j)$

else $C[i, j] := \infty$

$C[i, i] := 0$

for $k := 1$ to n do

for $i := 1$ to n do

for $j := 1$ to n do

if $C[i, k] + C[k, j] < C[i, j]$ then

$C[i, j] := C[i, k] + C[k, j]$

$X[i, j] := k$

After all distances are computed if least-cost-path from vertex i to j is desired:

output(i), call $path(i, j)$, output(j) where procedure $path(i, j)$

$k := X[i, j]$

if $k > 0$ then $\{path(i, k); output(k); path(k, j)\}$

Tracing the algorithm on the graph at p.41

Intermediate

C

X

Vertices

		1	2	3	4		1	2	3	4
{}	1	0	20	oo	oo	1	0	0	0	0
	2	oo	0	20	oo	2	0	0	0	0
	3	30	oo	0	10	3	0	0	0	0
	4	10	oo	oo	0	4	0	0	0	0
{1}		0	20	oo	oo		0	0	0	0
		oo	0	20	oo		0	0	0	0
		30	50	0	10		0	1	0	0
		10	30	oo	0		0	1	0	0
{1,2}		0	20	40	oo		0	0	2	0
		oo	0	20	oo		0	0	0	0
		30	50	0	10		0	1	0	0
		10	30	50	0		0	1	2	0

{1,2,3}

0	20	40	50
50	0	20	30
30	50	0	10
10	30	50	0

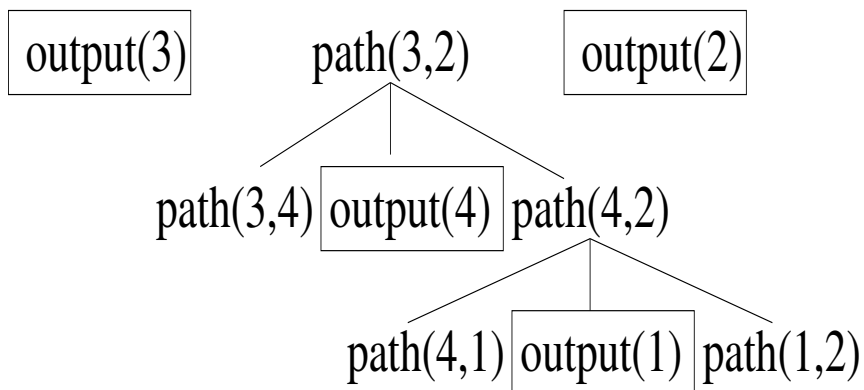
0	0	2	3
3	0	0	3
0	1	0	0
0	1	2	0

{1,2,3,4}

0	20	40	50
40	0	20	30
20	40	0	10
10	30	50	0

0	0	2	3
4	0	0	3
4	4	0	0
0	1	2	0

least-cost path from 3 and 2



output: 3 4 1 2

GREEDY: Current (local) greedy decision yields a correct solution

Example: Pack a given number of items with minimum total weight

Problem *myKnapsack*

input: a set of weights S , and a positive integer $N \leq |S|$

output: Minimum $\sum_{e \in S'} e$ such that $S' \subseteq S$, and $|S'| = N$

sum := 0

for $i := 1$ **to** N **do**

$m :=$ minimum element in S

$S := S - \{m\}$

$sum := sum + m$

return *sum*

Correctness Proof: Show that *myKnapsack* returns minimum subset sum over all subsets $S' \subseteq S$ with N elements

Let $e_1 \leq e_2 \leq \dots \leq e_N \leq \dots \leq e_{|S|}$ be the ascending order of elements in S . Clearly the algorithm returns $\sum_{e \in S'} e$ where $S' = \{e_1, e_2, \dots, e_N\}$. Is this optimum?

We will use proof by contradiction:

Suppose that the value returned by the algorithm is not optimum. Then there exists $S'' \subseteq S$ such that $|S''| = N$, and $\sum_{e \in S''} e < \sum_{e \in S'} e$. This requires that S'' contains an element e'' which IS NOT IN S' , and there exists an element e' in S' with $e'' < e'$ because otherwise sum of the elements in S'' cannot be smaller than that of in S' . This requires that $e'' < e_N$ because e_N is the largest element in S' . But such e'' must be included in S' since S' includes the smallest N elements. Contradiction! The assumption cannot be true. No such S'' exists. S' is optimal. The algorithm returns the optimum value.