Greedy Algorithms

Ex. Continuous Knapsack

As before we seek to

(1) Maximize $\sum_{i=1}^{n} x_i v_i$

(2) subject to $\sum_{i=1}^{n} x_i w_i \leq W$

where $W > 0$, $v_i > 0$, and $w_i > 0$ $(1 \leq i \leq n)$. However instead of $x_i \in \{0, 1\}$ we now allow $0 \leq x_i \leq 1$ for $1 \leq i \leq n$.

A vector $x = (x_1, \ldots, x_n)$ will be called a <u>feasible solution</u> if (2) is satisfied without regard to the optimality condition (1).

A <u>greedy strategy</u> consists of making a locally optimal (greedy) choice, then solving the subproblem arising from this choice.

In this problem that means including the "best" object which does not exceed the capacity constraint, then doing the same thing with the remaining objects and remaining capacity.

```
Knapsack (v, w, W)
1.) n ← length[v]
2.) for i ← 1 to n
3.)       x[i] ← 0
4.) weight ← 0
5.) while weight < W
*6.)      i ← THE "BEST" REMAINING OBJECT
7.)      if weight + w[i] ≤ W
8.)            x[i] ← 1
9.)            weight ← weight + w[i]
10.)           mark i AS INCLUDED.
11.)     else
12.)           x[i] ← (W - weight) / w[i]
13.)           weight ← W
14.) return x
```

GREEDY LOOP

THE "BEST" OBJECT IN (6) CAN BE INTERPRETED IN SEVERAL WAYS.

IN GENERAL WE DEFINE A <u>SELECTION FUNCTION</u> $f(i)$ WHICH ENCODES THE DESIRABILITY OF OBJECT i. LINE (6) THEN MAXIMIZES f OVER ALL REMAINING (UNMARKED) OBJECTS.

SOME POSSIBLE CHOICES FOR f IN THIS EXAMPLE ARE

- $f(i) = v_i$ (PROCESS OBJECTS IN ORDER OF DECREASING VALUES IN GREEDY LOOP.)
- $f(i) = \frac{1}{w_i}$ (INCREASING WEIGHTS)
- $f(i) = \frac{v_i}{w_i}$ (DECREASING VALUE TO WEIGHT RATIO.)

Ex. $n=5$, $W = 10$.

i	1	2	3	4	5
v_i	2	3	6.6	4	6
w_i	1	2	3	4	5
v_i/w_i	2	1.5	2.2	1	1.2

$f(i)$	x					VALUE
v_i	0	0	1	.5	1	14.6
$1/w_i$	1	1	1	1	0	15.6
v_i/w_i	1	1	1	0	.8	16.4

IN THIS EXAMPLE (AT LEAST) $f(i) = v_i/w_i$ GIVES THE BEST RESULT. IS THIS OPTIMUM?

THEOREM

IF WE MAXIMIZE v_i/w_i ON LINE (6) THEN KNAPSACK RETURNS AN OPTIMAL SOLUTION.

PROOF:

WITHOUT LOSS OF GENERALITY WE MAY ASSUME THE OBJECTS ARE INDEXED BY DECREASING v_i/w_i:

$$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \cdots \geq \frac{v_n}{w_n}$$

LET $x = (x_1, \ldots, x_n)$ BE THE SOLUTION RETURNED BY KNAPSACK.

IF ALL $x_i = 1$, THEN x IS CLEARLY OPTIMAL OTHERWISE LET j BE THE FIRST INDEX SUCH THAT $x_j < 1$. INSPECTING THE ALGORITHM IT'S CLEAR THAT

$$\begin{aligned} x_i &= 1 \qquad \text{FOR } 1 \leq i < j \\ x_j &< 1 \\ x_i &= 0 \qquad \text{FOR } j < i \leq n \end{aligned}$$

AND THAT

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

LET $V(x) = \sum_{i=1}^{n} x_i v_i$, THE VALUE OF x.
LET $y = (y_1, \ldots, y_n)$ BE ANY OTHER FEASIBLE SOLUTION, AND $V(y) = \sum_{i=1}^{n} y_i v_i$ ITS VALUE.

WE MUST SHOW $V(x) \geq V(y)$, AND HENCE x IS OPTIMAL.

SINCE y IS FEASIBLE $\sum_{i=1}^{n} y_i w_i \leq W$, AND THEREFORE

$$\sum_{i=1}^{n} (x_i - y_i) w_i = W - \sum_{i=1}^{n} y_i w_i \geq 0 .$$

NOW

$$\begin{aligned} V(x) - V(y) &= \sum_{i=1}^{n} (x_i - y_i) v_i \\ &= \sum_{i=1}^{n} (x_i - y_i) w_i \left(\frac{v_i}{w_i}\right) . \end{aligned}$$

OBSERVE THAT

$i < j \Rightarrow x_i = 1 \Rightarrow x_i - y_i \geq 0$ AND $\frac{v_i}{w_i} \geq \frac{v_j}{w_j}$

$$\therefore (x_i - y_i)\left(\frac{v_i}{w_i}\right) \geq (x_i - y_i)\left(\frac{v_j}{w_j}\right)$$

$i > j \Rightarrow x_i = 0 \Rightarrow x_i - y_i \leq 0$ AND $\frac{v_i}{w_i} \leq \frac{v_j}{w_j}$

$$\therefore (x_i - y_i)\left(\frac{v_i}{w_i}\right) \geq (x_i - y_i)\left(\frac{v_j}{w_j}\right)$$

$i = j \Rightarrow (x_i - y_i)\left(\frac{v_i}{w_i}\right) = (x_i - y_i)\left(\frac{v_j}{w_j}\right)$

Thus $(x_i - y_i)\left(\frac{v_i}{w_i}\right) \geq (x_i - y_i)\left(\frac{v_j}{w_j}\right)$ for $1 \leq i \leq n$.

$$\therefore\ V(x) - V(y) \geq \sum_{i=1}^{n} (x_i - y_i) w_i \left(\frac{v_j}{w_j}\right)$$

$$= \left(\frac{v_j}{w_j}\right) \sum_{i=1}^{n} (x_i - y_i) w_i$$

$$\geq 0 .$$

$\therefore$ $X = (x_1, \ldots, x_n)$ is optimal, as required. ///.

Examining loops 2-3 and 5-13, one sees that Knapsack runs in time $\Theta(n)$.

Our Dynamic Programming solution to the 0-1 Knapsack Problem ran in time $\Theta(nW)$, since the table was of size $n \times (W+1)$

Despite the 0-1 Knapsack Problem can be solved more efficiently using a Greedy strategy.

EX. SAME AS BEFORE, BUT NOW 0-1 KNAPSACK.

$n = 5, W = 10$

V	2	3	6.6	4	6
W	1	2	3	4	5
V/W	2	1.5	2.2	1	1.2

GREEDY SOLUTION:

$X = (1 \quad 1 \quad 1 \quad 1 \quad 0)$ { VALUE = 15.6, WEIGHT = 10

EXCERCISE

CHECK THAT DYNAMIC PROGRAM YIELDS THE VERY SAME SOLUTION.

EX. $n = 5, \quad W = 11$

V	1	6	18	22	28
W	1	2	5	6	7
V/W	1	3	3.6	3.67	4

GREEDY SOLUTION!

$X = (1 \quad 1 \quad 0 \quad 0 \quad 1)$ { VALUE = 35, WEIGHT = 10

EXCERCISE

CHECK THAT DYNAMIC PROGRAM YIELDS THE SOLUTION

$Y = (0 \quad 0 \quad 1 \quad 1 \quad 0)$ { VALUE = 40, WEIGHT = 11

THUS THE GREEDY STRATEGY DOES NOT ALWAYS YIELD THE OPTIMAL SOLUTION TO 0-1 KNAPSACK.

COIN CHANGING PROBLEM

AS BEFORE DENOMINATIONS $d = (d_1, \ldots, d_n)$ AND AN AMOUNT N TO BE DISBURSED WITH THE FEWEST COINS. (ASSUME AN UNLIMITED SUPPLY OF COINS IN EACH DENOMINATION.)

THE GREEDY STRATEGY TO COIN CHANGING IS AS FOLLOWS:

- FROM AMONGST ALL THE DENOMINATIONS WHOSE ADDITION WOULD NOT CAUSE THE SUM TO EXCEED N, CHOOSE THE LARGEST.
- STOP WHEN SUM IS N.

EXERCISE (HARD)

SHOW THAT FOR $d = (1, 5, 10, 25, 100)$ THE GREEDY STRATEGY YIELDS AN OPTIMAL SOLUTION FOR ANY $N \geq 0$.

EXERCISE (EASY)

SHOW THAT FOR $d = (1, 10, 25, 100)$ THE GREEDY STRATEGY DOES NOT YIELD AN OPTIMAL SOLUTION FOR SOME N. (e.g. $N = 30$)

EXERCISE (HARD)
CHARACTERIZE ALL DENOMINATION SETS $d=(d_1,..,d_n)$ SUCH THAT THE GREEDY STRATEGY YIELDS AN OPTIMAL SOLUTION FOR ALL $N \geq 0$.

HOW CAN WE TELL IF A GREEDY ALGORITHM WILL SOLVE A PARTICULAR OPTIMIZATION PROBLEM? IN GENERAL THIS IS A DIFFICULT QUESTION. THE KEY INGREDIENTS TO LOOK FOR ARE

- OPTIMAL SUBSTRUCTURE : OPTIMAL SOLUTIONS CONTAIN OPTIMAL SUBPROBLEM SOLUTIONS.

- GREEDY CHOICE PROPERTY : A GLOBALLY OPTIMAL SOLUTION CAN BE OBTAINED BY MAKING LOCALLY OPTIMAL (GREEDY) CHOICES (WITH RESPECT TO SOME SELECTION FUNCTION.)

MUCH OF THE HARD WORK IN DESIGNING A GREEDY ALGORITHM IS IN PROVING THAT THE GREEDY CHOICE PROPERTY IS SATISFIED.

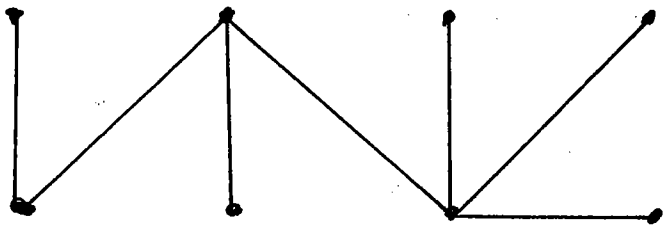
THE SITUATION IS COMPLICATED BY THE FACT THAT THE SELECTION FUNCTION MAY NOT COINCIDE WITH THE OBJECTIVE FUNCTION.

MINIMUM WEIGHT SPANNING TREES

WE RECALL SEVERAL DEFINITIONS RELATED TO AN (UNDIRECTED) GRAPH $G = (V, E)$.

- G IS CALLED <u>CONNECTED</u> IF G CONTAINS A u-v PATH FOR ALL $u, v \in V$.
- A <u>CYCLE</u> IN G IS A CLOSED PATH IN G. I.E. A PATH WHOSE INITIAL AND TERMINAL VERTICES ARE IDENTICAL
- G IS CALLED <u>ACYCLIC</u> IF IT CONTAINS NO CYCLES
- A GRAPH WHICH IS CONNECTED AND ACYCLIC IS CALLED A <u>TREE</u>.

<u>Ex.</u>



$|V| = 8$, $|E| = 7$

THEOREM

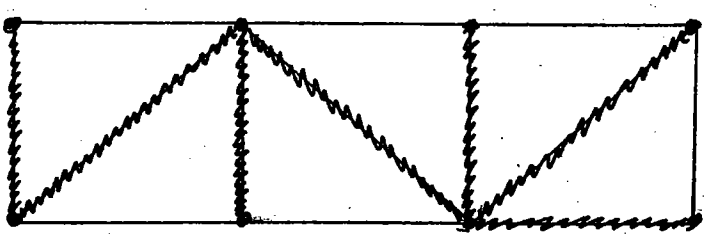
THE FOLLOWING ARE EQUIVALENT.

(1) G is A TREE
(2) G is CONNECTED AND $|E| = |V| - 1$
(3) G is ACYCLIC AND $|E| = |V| - 1$
(4) G is ACYCLIC, BUT IF A NEW EDGE is ADDED, A UNIQUE CYCLE is CREATED.
(5) G is CONNECTED, BUT THE REMOVAL OF ANY EDGE DISCONNECTS G.
(6) THERE is A UNIQUE u-v PATH in G FOR ALL $u, v \in V$.

FOR A PROOF SEE APPENDIX B P. 1085, OR TAKE CE 177.

- A SUBGRAPH OF G is CALLED A SPANNING TREE IF IT IS A TREE, AND IT INCLUDES ALL VERTICES OF G

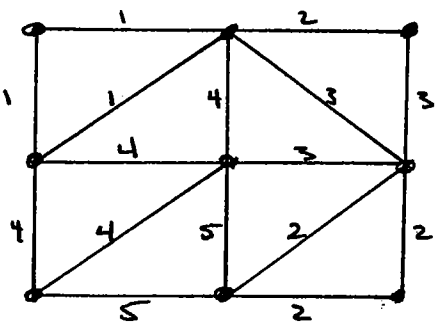
EX.



SUPPOSE $G = (V, E)$ IS EQUIPPED WITH A NON-NEGATIVE WEIGHT FUNCTION ON EDGES

$$w : E \longrightarrow \mathbb{R}_+$$

EX.



- THE <u>WEIGHT</u> OF A SPANNING TREE IS THE SUM OF THE WEIGHTS OF ITS EDGES.

<u>PROBLEM</u>

DETERMINE A SPANNING TREE OF MINIMUM WEIGHT IN G.

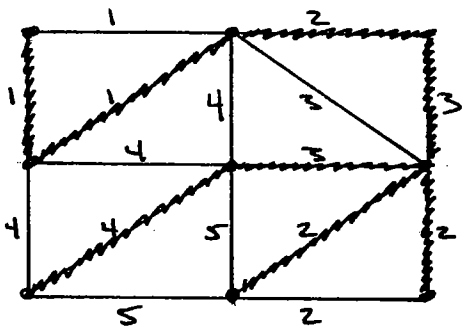
WE EXAMINE TWO FAMOUS GREEDY ALGORITHMS WHICH SOLVE THIS PROBLEM.

IN WHAT FOLLOWS LET $G = (V, E)$, $|V| = n$.

PRIM'S ALGORITHM (23.2)

- CHOOSE AN INITIAL VERTEX (WHICH IS A TREE)
- AMONGST ALL EDGES INCIDENT WITH THE CURRENT TREE WHOSE ADDITION WOULD NOT CREATE A CYCLE, CHOOSE ONE OF MINIMUM WEIGHT.
- STOP WHEN N-1 EDGES HAVE BEEN SELECTED.

EX



$W(T) = 18$

OBSERVE THAT AT EACH STAGE OF EXECUTION, PRIM'S ALGORITHM MAINTAINS A TREE SINCE NO CYCLES ARE CREATED AND ONLY INCIDENT EDGES ARE ADDED.

WHEN THIS TREE CONTAINS $n-1$ EDGES IT MUST HAVE n VERTICES (BY PREVIOUS THEOREM), HENCE IT IS A SPANNING TREE.

THEOREM

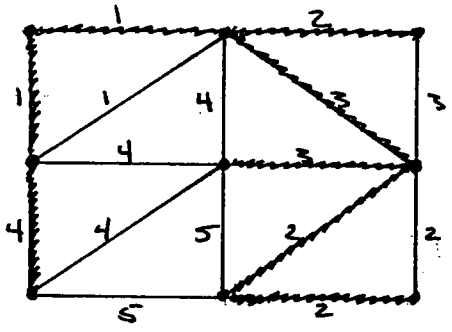
THIS SPANNING TREE HAS MINIMUM POSSIBLE WEIGHT.

(SEE BOOK OR TAKE CE 177 FOR PROOF.)

KRUSKAL'S ALGORITHM (23.2)

- CHOOSE AN EDGE OF MINIMUM WEIGHT
- AMONGST ALL EDGES WHICH DO NOT CREATE A CYCLE WITH PREVIOUSLY SELECTED EDGES, CHOOSE ONE OF MINIMUM WEIGHT.
- STOP WHEN $n-1$ EDGES HAVE BEEN SELECTED.

EX.



$W(T) = 18$

OBSERVE THAT AT EACH STAGE OF EXECUTION KRUSKAL'S ALGORITHM HAS CREATED A FOREST (UNION OF DISJOINT SUBTREES) SINCE NO CYCLES ARE CREATED.

WHEN THIS FOREST CONTAINS $n-1$ EDGES IT MUST ALSO HAVE n VERTICES. (ANY FOREST ~~GRAPH~~ WITH $n-1$ EDGES HAS AT LEAST n VERTICES. THIS FOREST CAN CONTAIN NO MORE THAN n VERTICES SINCE IT IS A SUBGRAPH OF G.)

THUS THE RESULTING FOREST IS CONNECTED (BY PREVIOUS THEOREM) AND IS A SPANNING TREE IN G.

THEOREM

THIS SPANNING TREE HAS MINIMUM WEIGHT AMONG ALL SPANNING TREES IN G.

PROOF.

LET T BE THE SPANNING TREE IN G CREATED BY KRUSKAL'S ALGORITHM, AND LET S BE ANY OTHER SPANNING TREE. WE MUST SHOW

$$w(T) \leq w(S)$$

LET $e_1, e_2, \ldots e_{n-1}$ BE THE EDGES OF T IN THE ORDER SELECTED BY KRUSKAL'S ALGORITHM. SINCE $S \neq T$ THERE IS A FIRST EDGE e_k WHICH IS NOT IN S, I.E.

$$\{e_1, \ldots, e_{k-1}\} \subseteq E(S)$$
$$e_k \notin E(S)$$

LET H BE THE SUBGRAPH OBTAINED BY ADDING e_k TO S: $H = S + e_k$. BY THE TREENESS THEOREM H CONTAINS A UNIQUE CYCLE WHICH INCLUDES e_k, CALL IT C. NOTE C MUST CONTAIN AN EDGE e OF S WHICH IS NOT IN T, FOR OTHERWISE C IS CONTAINED IN THE ACYCLIC T.

NOW REMOVE e FROM H TO OBTAIN A SUBGRAPH R, WHICH IS CONNECTED SINCE e BELONGS TO A CYCLE IN H.

$$R = H - e = S + e_k - e$$

SINCE R IS CONNECTED AND HAS $n-1$ EDGES, IT IS ANOTHER SPANNING TREE OF G, BY TREENESS THEOREM.

THE NATURE OF KRUSKAL'S ALGORITHM GUARANTEES THAT $w(e_k) \leq w(e)$.

Pf: e DOES NOT FORM A CYCLE WITH $\{e_1, \ldots, e_{k-1}\}$ SINCE $\{e_1, \ldots, e_{k-1}, e\} \subseteq E(S)$. THUS IF $w(e) < w(e_k)$, THEN KRUSKAL WOULD HAVE CHOSEN e ON THE k^{TH} ITERATION OF THE GREEDY LOOP INSTEAD OF e_k. //

THUS R IS A SPANNING TREE OF G WITH ONE MORE EDGE IN COMMON WITH T THAN S, AND SATISFYING $w(R) \leq w(S)$.

IF $R = T$ WE ARE DONE, OTHERWISE WE MAY PERFORM THIS SAME CONSTRUCTION WITH R IN PLACE OF S.

i.e. CONSTRUCT ANOTHER SPANNING TREE R_2 WITH ONE MORE EDGE IN COMMON WITH T THAN R, AND SATISFYING $W(R_1) \leq W(R)$.

CONTINUING IN THIS FASHION WE CONSTRUCT A SEQUENCE OF SPANNING TREES WHICH MUST EVENTUALLY REACH T:

$$W(T) \leq \ldots \leq W(R_1) \leq W(R) \leq W(S),$$

SO $W(T) \leq W(S)$ AS REQUIRED.

///

MATROIDS AND THE GREEDY ALGORITHM

A MATROID IS AN ABSTRACT MATHEMATICAL STRUCTURE WHICH GENERALIZES MANY EXAMPLES WHERE A GREEDY STRATEGY APPLIES.