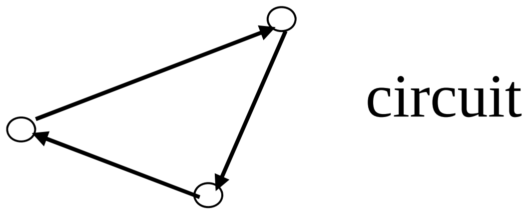
2.3.4 Optimal paths in directed acyclic graphs

Definition: A directed graph $G = (N, A)$ is *acyclic* if it contains no circuits. A directed acyclic graph is referred to as *DAG*.



Problem

Given a directed acyclic $G = (N, A)$ with a cost $c_{ij} \in \mathbb{R}$ for each $(i, j) \in A$, and nodes s and t, determine a shortest (longest) path from s to t.

Property

The nodes of any directed acyclic graph G can be ordered topologically, that is, indexed so that

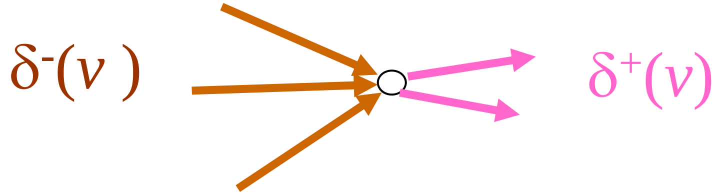
for each arc $(i, j) \in A$ we have $i < j$.

Topological order

Such a topological order can be exploited in a very efficient Dynamic Programming algorithm to find shortest (or longest) paths in directed acyclic graphs.

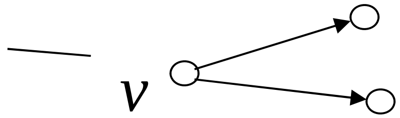
Topological ordering method

$G = (N, A)$ is represented via the lists of predecessors $\delta^-(v)$ and successors $\delta^+(v)$ for each node v



1. Assign the smallest positive integer not yet assigned to a node $v \in N$ with $\delta^-(v) = \emptyset$.

such a node *v* exists because *G* does not contain circuits

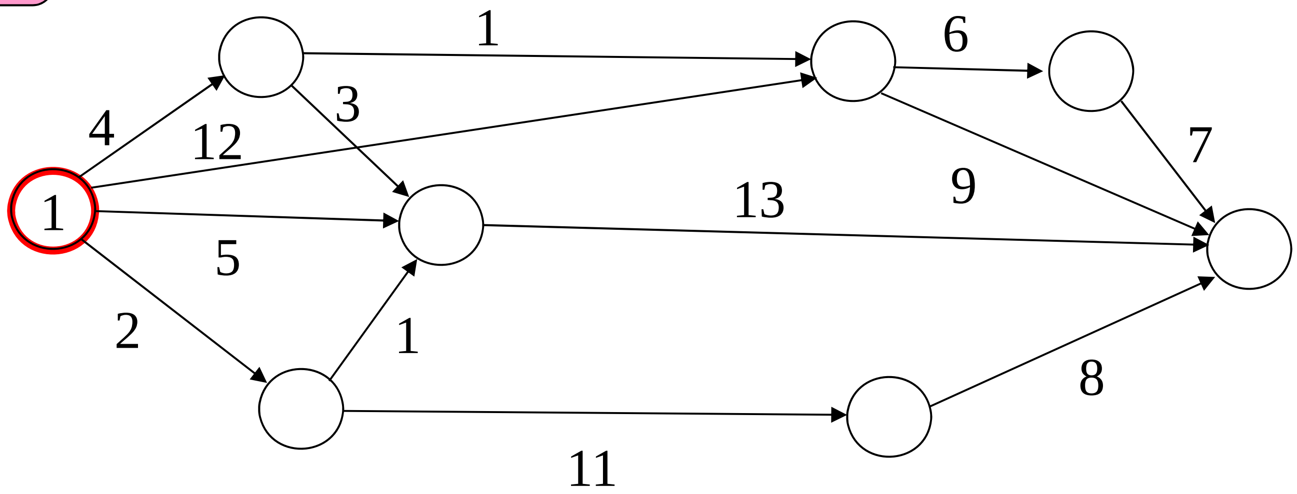


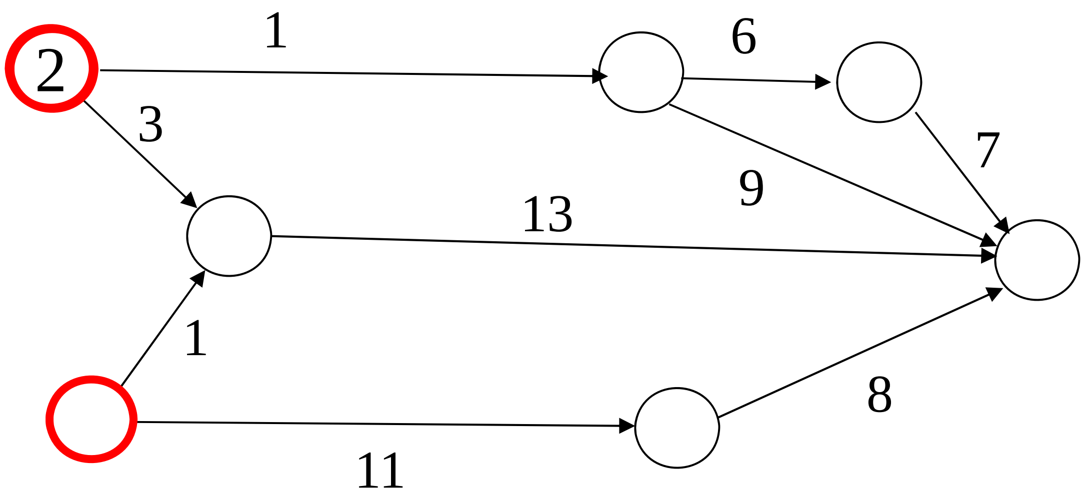
2. Delete the node *v* with all its incident arcs.

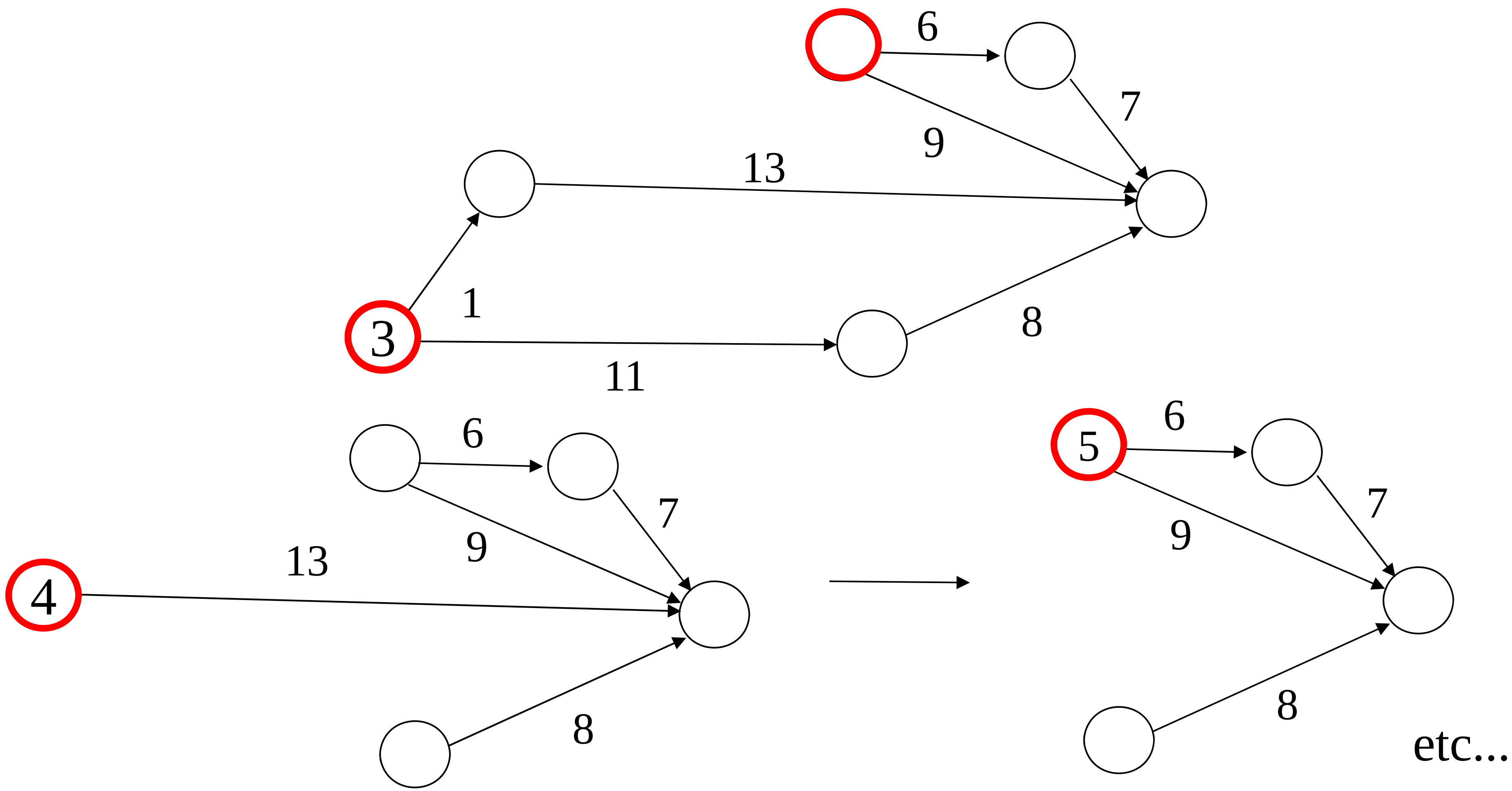
3. Go back to 1. until there are nodes in the current subgraph.

Example

G



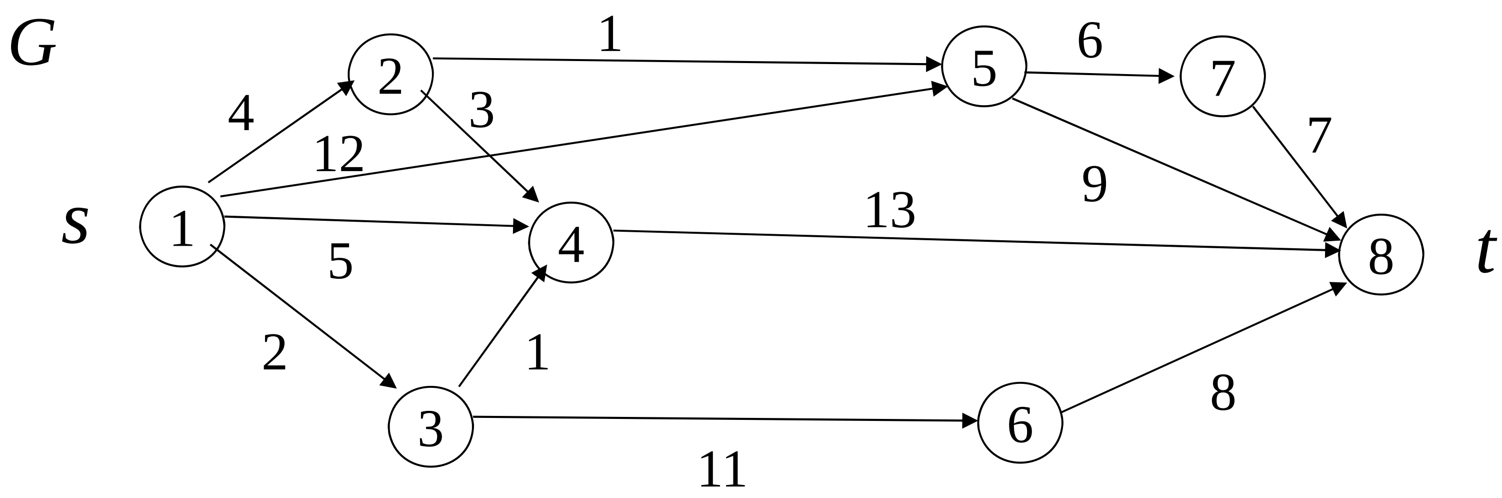




Complexity: $O(m)$ where $m=|A|$

because each node and each arc is considered at most once.

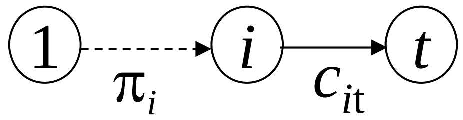
Example



Given *G* and a topological order of the nodes, find a shortest path from node 1 to node 8 ?

Dynamic Programming for shortest paths in DAGs

Any shortest path $\pi_t : 1 \dashrightarrow t$ with at least two arcs can be subdivided into two parts π_i and (i,t):



where $\pi_i : 1 \dashrightarrow i$ is a shortest subpath from s to i.

Definition: For each node $i = 1,\ldots, t$

$$L(i) = \underline{cost} \text{ of a } \underline{shortest\ path} \text{ from } 1 \text{ to } i.$$

Clearly

$$L(t) = \min_{(i,t)\in\delta^-(t)} \{L(i) + c_{it}\}$$

where the minimum is taken over all possible predecessors i of t in a path from 1 to t.

If G is directed and acyclic and the nodes are topologically ordered, the only possible predecessors of t in a shortest path π_t from 1 to t are those with index $i < t$ and hence:

$$L(t) = \min_{i<t}\{L(i) + c_{it}\}$$

Observation: In a graph containing circuits, any node different from t can be a predecessor of t in π_t!

For DAGs whose nodes are topologically ordered

$L(t\text{-}1)$,.... and $L(1)$ satisfy the same type of recursive relations:

$$L(t\text{-}1) = \min_{i<t-1}\{L(i) + c_{i(t-1)}\}$$

$$\vdots$$

$$L(2) = \min_{i=1}\{L(i) + c_{i2}\} = L(1) + c_{12}$$

$$L(1) = 0$$

which can be solved in the reverse order:

$$L(1) = 0$$

$$L(2) = L(1) + c_{12}$$

$$\vdots$$

$$L(t) = \min_{i<t}\{L(i) + c_{it}\}$$

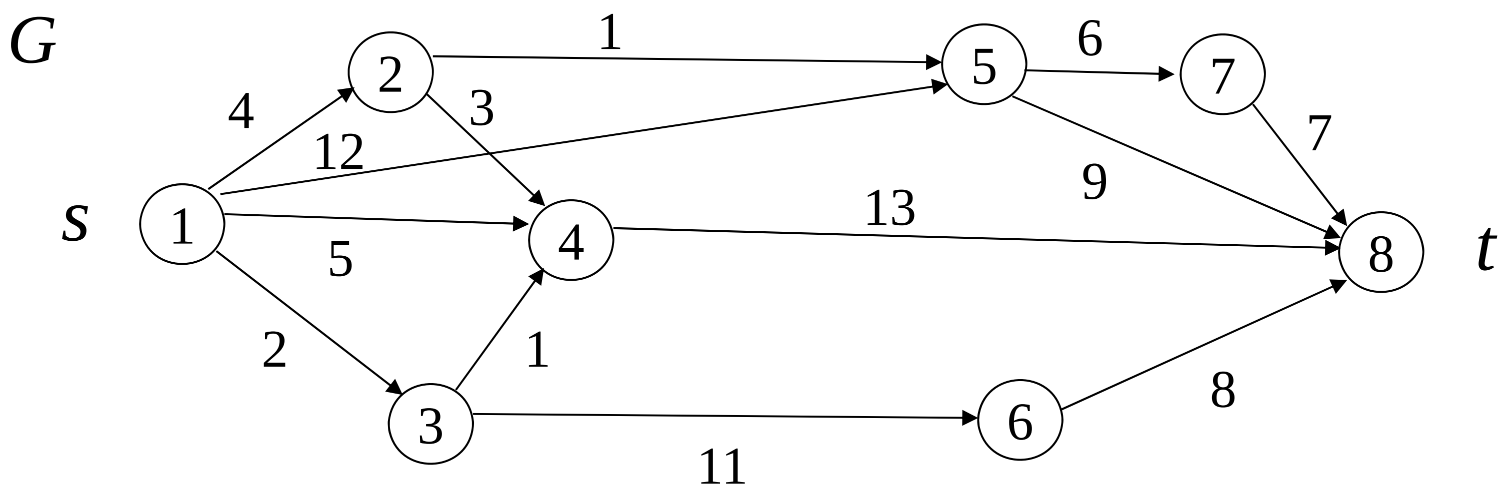
Example

$L(1) = 0$ pred(1)=1

$L(2) = L(1) + c_{12} = 0 + 4 = 4$ pred(2)=1

$L(3) = L(1) + c_{13} = 0 + 2 = 2$ pred(3)=1

$$L(4) = \min_{i=1,2,3}\{L(i) + c_{i4}\} = \min\{0 + 5, 4 + 3, 2 + 1\} = 3$$ pred(4)=3



[.] indicates the predecessor of node i in a shortest path from 1 to i

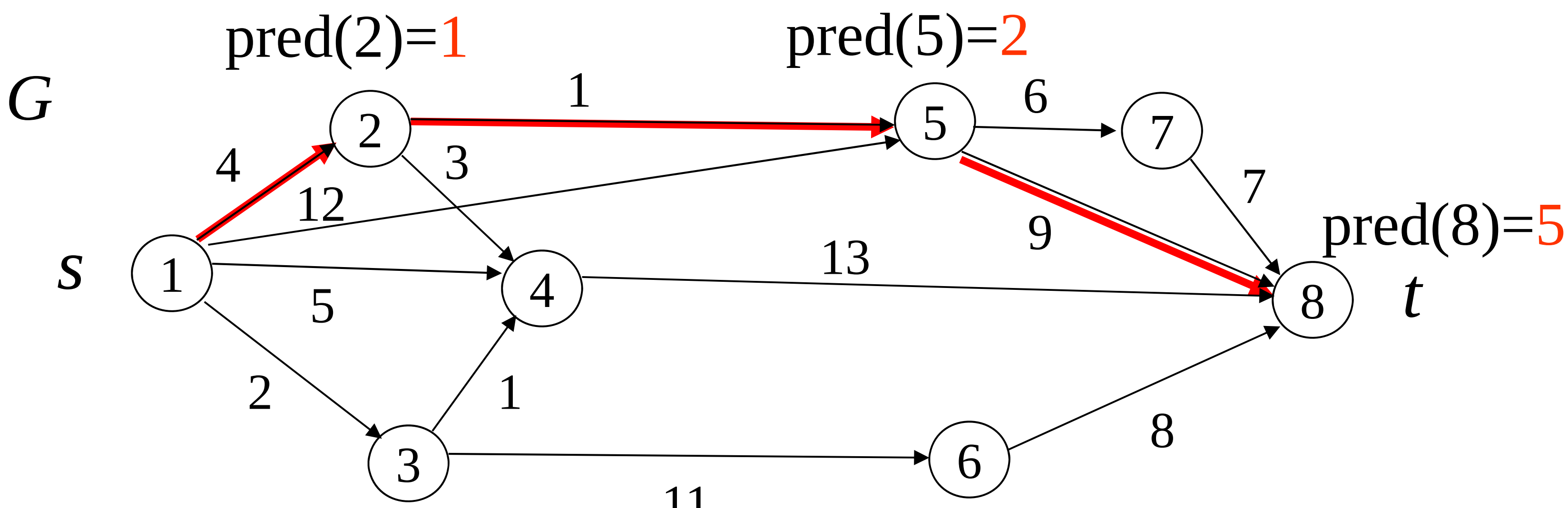
$$L(5) = \min_{i=1,2}\{L(i) + c_{i5}\} = \min \{ 0 + 12, 4 + 1 \} = 5$$ pred(5)=2

$$L(6) = \min_{i=3}\{L(i) + c_{i6}\} = \min \{ 2 + 11 \} = 13$$ pred(6)=3

$$L(7) = L(5) + c_{57} = 5 + 6 = 11$$ pred(7)=5

$$L(8) = \min_{i=4,5,6,7}\{L(i) + c_{i8}\} = \min \{ 3+13, 5+9, 13+8, 11+7 \} = 14$$

pred(8)=5



shortest path from 1 to 8 of total cost 14

Complexity DP for shortest paths in DAGs

```
BEGIN
  Sort the nodes of G topologically;
  L[1] := 0;
  FOR j:=2 TO n DO    /* in topological order */
    L[j] := min {L[i] + c_ij : (i,j) ∈ δ⁻(j), i < j};
    pred[j] := v with
        (v,j) =  argmin {L[i] + c_ij : (i,j) ∈ δ⁻(j), i < j};
  END-FOR
END
```

- Topological ordering of the nodes: $O(m)$ with $m=|A|$
- Each node and arc are considered exactly once: $O(n+m)$

Overall **complexity**: $O(m)$

Adaptation of DP for longest paths in DAGs

Straightforward adaptation of the Dynamic Programming method to find longest paths in directed acyclic graphs:

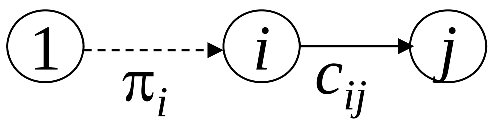
$$L(\mathrm{t}) = \max_{i<t}\{L(i) + c_{it}\}....$$

Proposition: The Dynamic Programming algorithm for finding shortest (longest) paths in DAGs is exact.

This is due to:

Optimality principle

For any shortest (longest) path π_j : $1 \dashrightarrow j$ there exists $i < j$ such that we have the decomposition:



where the subpath π_i : $1 \dashrightarrow i$ is of minimum (maximum) length from s to i.

Dynamic Programming (DP)

Proposed in 1953 by Richard Bellman (1920-1984)

General technique in which an optimal solution, composed of a sequence of elementary decisions, is determined by solving a set of recursive equations.

DP is applicable to any "sequential decision problem" for which the optimality property is satisfied.

A wide range of applications: optimal control (rocket launch), equipment maintenance and replacement, selection of inspection points along a production line,…

Exercise

A company buys a new machine for 12 K€.

Annual maintenance costs for the next five years:

Age (years)	Maintenance (K€)	Recover (K€)
0	2	-
1	4	7
2	5	6
3	9	2
4	12	1

To avoid high maintenance costs of an older machine, the machine can be replaced at the beginning of the 2^{nd}, 3^{rd}, 4^{th} or 5^{th} year with a new machine.

To simplify the model, suppose that the cost of a new machine is constant (12 K€).

Objective: minimize the *net total cost* (price + maintenance – recover) over the 5 year horizon.

Age	Main.	Reco.
0	2	-
1	4	7
2	5	6
3	9	2
4	12	1

If we buy a machine at beginning of 1st year and sell it at the beginning of 2nd year, the net cost is: 12 + 2 - 7 = 7

If we buy it at the beginning of 1st year and sell it at the beginning of 3rd year, the net cost is:12 + 2 + 4 - 6 = 12

etc...

Show how the problem of determining an optimal maintenance - replacement policy of minimum total cost (for the next 5 years) can be solved via Dynamic Programming.

Hint: Reduce it to a minimum cost path problem in an appropriate acyclic graph.

Find all the optimal maintenance-replacement policies.