Algorithms

DATE: 5/5/22

KNAPSACK

- we have n jobs, the ith of which takes w_i seconds to finish, and has value v_i.

  Our CPU is available for W seconds.

  A set S of jobs is feasible if:

$$\sum_{i \in S} w_i \leq W$$

What is a set S of jobs satisfying $\sum_{i \in S} w_i \leq W$ and having maximum total value? $\sum_{i \in S} v_i$

We assume that $W, w_1, \ldots, w_n$ are integers.

$$OPT(i, W) = \max_{S \subseteq \{1,2,\ldots,i\}} \sum_{j \in S} v_j$$

$$\sum_{j \in S} w_j \leq W$$

w_j = "length" of job j
v_j = "payment" you get for running job j

T: If $O_{i,w}$ is an optimal solution (a set of jobs) for the jobs $\{1, \ldots, i\}$ and with w available seconds. Then,

- If $i \notin O_{i,w}$, then $OPT(i,w) = OPT(i-1, w)$, and
- If $i \in O_{i,w}$, then $OPT(i,w) = v_i + OPT(i-1, w-w_i)$

Moreover, if $w_i > w$, then $i \notin O_{i,w}$, thus $OPT(i,w) = OPT(i-1, w)$

KNAPSACK (n, W)

INITIALIZE THE ARRAY $M[0 \ldots n][0 \ldots W]$ $O(nW)$

LET $M[0][W] = 0$ $\forall\, 0 \leq w \leq W$ $O(W)$

FOR $i = 1 \ldots n$

  FOR $w = 0 \ldots W$

$O(n \cdot W)$

IF $(w_i > W)$ or $(M[i-1][W] > v_i + M[i-1][W-w_i])$

$M[i][W] = M[i-1][W]$

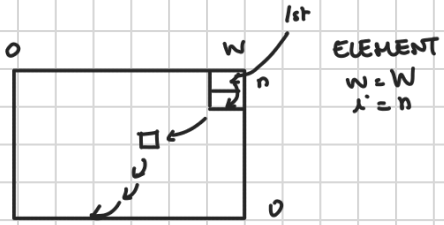
ELSE:

$M[i][W] = v_i + M[i-1][W-w_i]$

RETURN M

We could run $M = KNAPSACK(n, W)$, and then conclude that $M[n][W]$ is the value of the best solution

FINDSOL (n, W)



M = KNAPSACK (n, W) $O(n \cdot W)$

S = [ ]

w = W

i = n

WHILE $i > 0$:

IF $M[i][W] == M[i-1][W]$

$i -= 1$ // throw away i

ELSE:

S.APPEND (i) // schedule i

$W -= w_i$

$i -= 1$

$O(n)$

IF $M[i][W] \neq M[i-1][W]$:

S.APPEND (i)

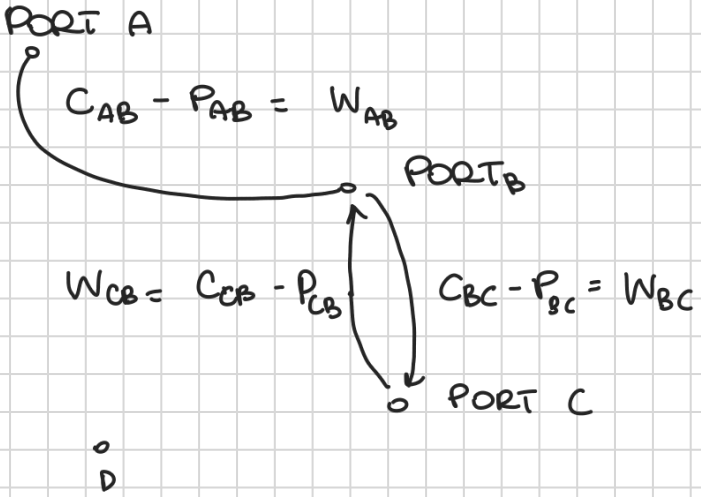
$W -= w_i$

$i -= 1$

RETURN S // $\sum_{j \in S} v_j = M[n][W]$ and $\sum_{j \in S} w_j \leq W$

SHORTEST PATHS ON GRAPHS WITH "NEGATIVE AND/OR POSITIVE" WEIGHTS.

cruise ship / taxi drive

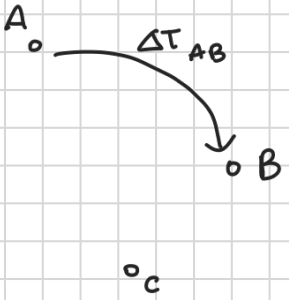


$W_{AB} > 0 \quad C_{AB} > P_{AB}$

$W_{AB} = 0 \quad C_{AB} = P_{AB}$

$W_{AB} < 0 \quad C_{AB} < P_{AB}$

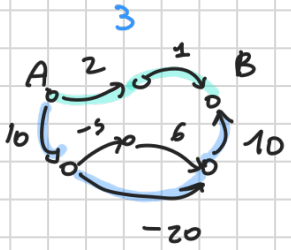
CHEMICAL REACTIONS



$V = \{A, B, C\}$ chemical compounds

HOW to go from s to t with the smallest increase in temperature

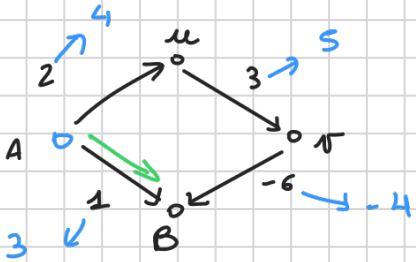
---

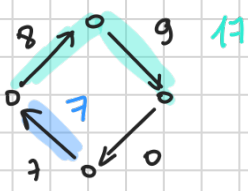


A TO B

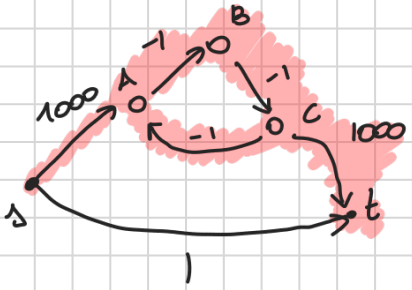
DIJKSTRA DOESN'T WORK HERE

A	0
B	1
u	2





ADD $-\min_e W_e$ to each edge (so that edge weights become non-neg.)



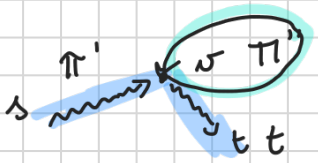
we assume that the graph has no negative cycles

L1: IF G is a graph with no negative cycles, then ∃ exists a shortest path from s to t that is simple (it has no node repetition), and that contains $\leq n-1$ edges.

P: Let π be a s-t shortest path with minimal number of edges.

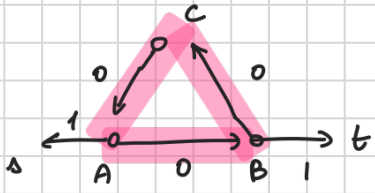
suppose that $\exists\, v$ s.t. $\pi = \pi' v \pi'' v \pi'''$

Since G contains no negative cycles, it must be that $v\pi''$ is a cycle having a positive total edge weight.



Consider, now, the path $\tilde{\pi} = \pi' v \pi''$. This path has fewer edges than π. Moreover the total weight of $\tilde{\pi}$ is no larger than the total weight of π. Since π has more edges than $\tilde{\pi}$, and since the total weight of $\tilde{\pi}$ is not larger than that of π, π cannot be a s-t shortest path with a minimal numbers of edges. CONTRADICTION.

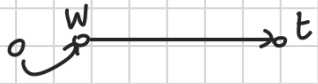
Then no node is repeated in a shortest path with a minimal number of edges — since we have n nodes, such a path has at most $n-1$ edges ■



$w(s\ AB\ t) = 2$
$w(s\ ABCAB\ t) = 2$

Let $OPT(i, v)$ be the minimum cost/weight of a path from v to t, having **at most** i edges.

Let π be a path of i edges from v to t, having minimum cost $OPT(i, v)$



π has at most i edges

- If π has at most $i-1$ edges, then $OPT(i,v) = OPT(i-1,v)$
- If π has exactly i edges, then $\exists w$ such that π starts from v, moves to w, and then progresses optimally to t using at most $i-1$ edges. That is,

$$OPT(i,v) = c_{vw} + OPT(i-1, w)$$

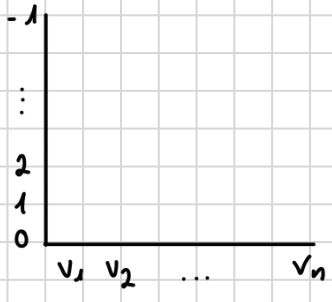
This DP is called "the BELLMAN-FORD" algorithm.

$$OPT(0, t) = 0$$
$$OPT(0, v) = \infty \quad \forall v \neq t$$

L2: If $i > 0$, $OPT(i, v) = \min\left(OPT(i-1, v),\ \min_{w \,:\, (v,w) \in E} \left(c_{vw} + OPT(i-1, w) \right) \right)$ (and)

BELLMAN-FORD (G, s, t):

```
INITIALIZE M[0 ... n-1][v ∈ V(G)]
M[0][t] = 0
M[0][v] = +∞   ∀ v ∈ V - {t}      (FOR EACH NODE DIFFERENT FROM T ITSELF.)
FOR i = 1 .... n-1:
    FOR v ∈ V(G):
        M[i][v] = M[i-1][v]
        FOR w ST. (v,w) ∈ E(G):
            γ = c_{v,w} + M[i-1][w]
            IF γ < M[i][v]:
                M[i][v] = γ
RETURN M   (or M[n-1][s])
```



RECURRENCE OF L2

THM: Bellman-Ford returns the shortest distance from s to t if the graph contains no no negative cycles.

P: Apply L1 and L2.

How can you deal with negative cycles if they exist?

If we let the algo go for an extra iteration, so instead of stoping at $n-1$, we let It stop at n, then we can observe and proof the following lemma: if the OPT values with $i=n$ coincide to the OPT values with $i=n-1$, no negative cycles exists, otherwise some negative cycles exists.
It may happen that a negative cycle exists but can't be reached.

L'algoritmo di Bellman-Ford è stato il primo algoritmo ad introdurre l'approccio dinamico e risolve il problema dei cammini minimi da sorgente unica nel caso generale in cui i pesi degli archi possono essere negativi.
Dato un grafo orientato pesato $G=(V, E)$ con sorgente s e funzione peso $w: E \to \mathbb{R}$, l'algoritmo di Bellman-Ford restituisce un valore booleano che indica se esiste oppure no un ciclo di peso negativo che è raggiungibile dalla sorgente. Se un tale ciclo esiste, l'algoritmo indica che il problema non ha soluzione. Se un tale ciclo non esiste, l'algoritmo fornisce i cammini minimi e i loro pesi.