Floyd-Warshall

All pairs shortest paths $\quad G=(V,E)$

$V = v_1, v_2 \dots v_n$

For each pair of vertices a, b

for each $k = 0, \dots, n$

$d_k(a,b)$ = length of shortest path from a, b if we are not allowed to use vertices $v_{k+1}, \dots, v_n$ as intermediate steps in path

→ $k=0$ $\quad d_0(a,b)$ easy to compute

→ if we know $d_k(a,b)$ for all a,b it easy to compute $d_{k+1}(a,b)$ for all a,b

→ $k=n$ $\quad d_n(a,b)$ solution to our original problem

Dynamic Programming

$$d_0(a,b) = \begin{cases} 0 & a = b \\ \ell(a,b) & (a,b) \in E \\ \infty & \text{otherwise} \end{cases}$$

$$\boxed{\begin{aligned} d_{k+1}(a,b) = \min\{\, & d_k(a,b), \\ & d_k(a, v_{k+1}) + d_k(v_{k+1}, b) \,\} \end{aligned}}$$

$$\overbrace{a \rightsquigarrow v_{k+1}}^{d_k(a, v_{k+1})} \overbrace{\rightsquigarrow b}^{d_k(v_{k+1}, b)}$$

not contain v_{k+1} — not contain v_{k+1}

$$d_n(a,b) = d(a,b)$$

construct $dist[k, a, b]$ so that $= d_k(a,b)$

for $a, b \in V$

$$\text{dist}[0, a, b] = \begin{cases} 0 & \text{if } a = b \\ \ell(a,b) & \text{if } (a,b) \in E \\ \infty & \text{otherwise} \end{cases}$$

for $k = 1$ to n:

for a in V:

for b in V:

$\text{dist}[k, a, b] =$

$\min(\text{dist}[k-1, a, b],$

$\text{dist}[k-1, a, v_k] + \text{dist}[k-1, v_k, b])$

return $\text{dist}[n::]$

At the end of iteration k of for loop

$\forall a, b \quad \text{dist}[k, a, b] = d_k(a, b)$

$k = 0$ ✓

true for k

$\text{dist}[k+1, a, b]$

$= \min(\text{dist}[k, a, b],$
$\text{dist}[k, a, v_{k+1}] + \text{dist}[k, v_{k+1}, b])$

$= \min(d_k(a,b), \; d_k(a, v_{k+1}) + d_k(v_{k+1}, b))$

$= d_{k+1}(a, b)$

Edit Distance

given 2 strings: A = "past"

B = "arts"

change A into B with a sequence of operations

each operation can be

- delete one letter
- insert a letter
- change a letter

do as few operation as possible

~~p~~ast → ast → art → arts

delete p — change s → r — add s

p a s t ␣

␣ a r t s

string alignment

given $A = a_1 \ldots a_n$

$B = b_1 \ldots b_m$

definition

$ED(i,j) =$ edit distances between $a_1 \ldots a_i$ and $b_1 \ldots b_j$

$ED(0, j) = j$

$ED(i, 0) = i$

easy

$$ED(i,j) = \begin{cases} 1 + ED(i-1, j) & (1) \\ 1 + ED(i, j-1) & (2) \\ ED(i-1, j-1) + \begin{cases} 0 & \text{if } a_i = b_j \\ 1 & \text{if } a_i \neq b_j \end{cases} & (3) \end{cases}$$

(3) $\begin{array}{l} \ldots\ldots \\ \ldots\ldots \end{array} \left| \begin{array}{l} a_i \\ b_j \end{array} \right|$

$\underbrace{a_1 \ldots a_{i-1}}\, a_i \to \ldots \to \underbrace{b_1 \ldots b_{j-1}}\, a_i \to b_1 \ldots b_{j-1} b_j$

$ED(n, m)$ what we want to compute

```
def edit_distance (A, B):
    n = len(A)
    m = len(B)
    D = (n+1)×(m+1) array of integers
    for i = 0 to n:
        D[i, 0] = i
    for j = 0 to m
        D[0, j] = j

n   for i = 1 to n:
m       for j = 1 to m:
            D[i, j] = min (
                        D[i-1, j] + 1,
O(1)                    D[i, j-1] + 1,
                        D[i-1, j-1] + (1 if A[i] ≠ A[j]
                                       0 else)
                        )
    return D[n, m]
```

$O(n \cdot m)$

$len(A) = len(B) = n$

edistance K

$a_1 \sqcup a_2 a_3 \sqcup - - \sqcup a_n$

$b_1 \sqcup - - \sqcup b_2 \quad — b_n \sqcup$

$$\binom{n+K}{K} \cdot \binom{n+K}{K}$$

bocconi

boccon

bocron

bacron

macron

Knapsack

bag in which you can put items of total weight $\leq B$ integer

set of items

item	weight	cost
1	w_1	c_1
2	w_2	c_2
	$\vdots$	
n	w_n	c_n

w_i integer

goal: find subset $S \subseteq \{1 \ldots n\}$

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

$$\sum_{i \in S} c_i \text{ is maximized}$$

ex $\quad B = 5$

	weight	cost
1	4	8
2	5	7
3	2	4
4	2	5

item	weight	cost
1	3	2
2	3	2
3	4	3

$B = 6$

(1) item n is chosen

opt : $c_n + ($ opt of choosing from items $1 \ldots n-1$ with weight bound $B - w_n)$

(2) item n is not chosen

opt : opt of choosing from items $1 \ldots n-1$ with weight bound B

$K(b, i) =$ optimal solution for problem in which we can only use a subset of items $1, \ldots, i$ weight bound is b

$K(0, i) = 0$
$K(b, 0) = 0$

$$K(b,i) = \max\Big(\overbrace{c_i + K(b-w_i, i-1)}^{\text{valid if } w_i \leq b},\ K(b, i-1) \Big)$$

$K(B, n)$ is problem we want to solve

```
def knapsack(n, w, c, B):
    K (B+1) x (n+1) array
    for i = 0 to n:
        K[0, i] = 0
    for b = 0 to B:
        K[b, 0] = 0
    for b = 1 to B:
        for i = 1 to n:
            K[b, i] = K[b, i-1]
            if w[i] ≤ b and
                c[i] + K[b-w[i], i-1]
                > K[b, i]:
              K[b, i] = c[i] + K[b-w[i], i-1]

    return K[B, n]
```

$O(B \cdot n)$