

Dynamic Programming

All pair shortest paths.

Dijkstra $O(V^2) \rightarrow O(V^3)$ for all pairs.

$D[i, j]$ = shortest path from i to j

$D_k[i, j]$ = shortest path from i to j
only using vertices $1, \dots, k$ as
intermediate nodes

$D_0[i, j] = \infty$ if no edge (i, j)

d_{ij} = length of edge from i to j otherwise

$V = \{1, \dots, n\}$

recurrence for $D_k[i, j] = \min \left[D_{k-1}[i, j], D_{k-1}[i, k] + D_{k-1}[k, j] \right]$

call 1 :

call 2 :

keep two
2D arrays

for
each
 k
update
2D
array

array $D = (d_{ij})$

for $k = 1$ to n do

for $i = 1$ to n do

for $j = 1$ to n do

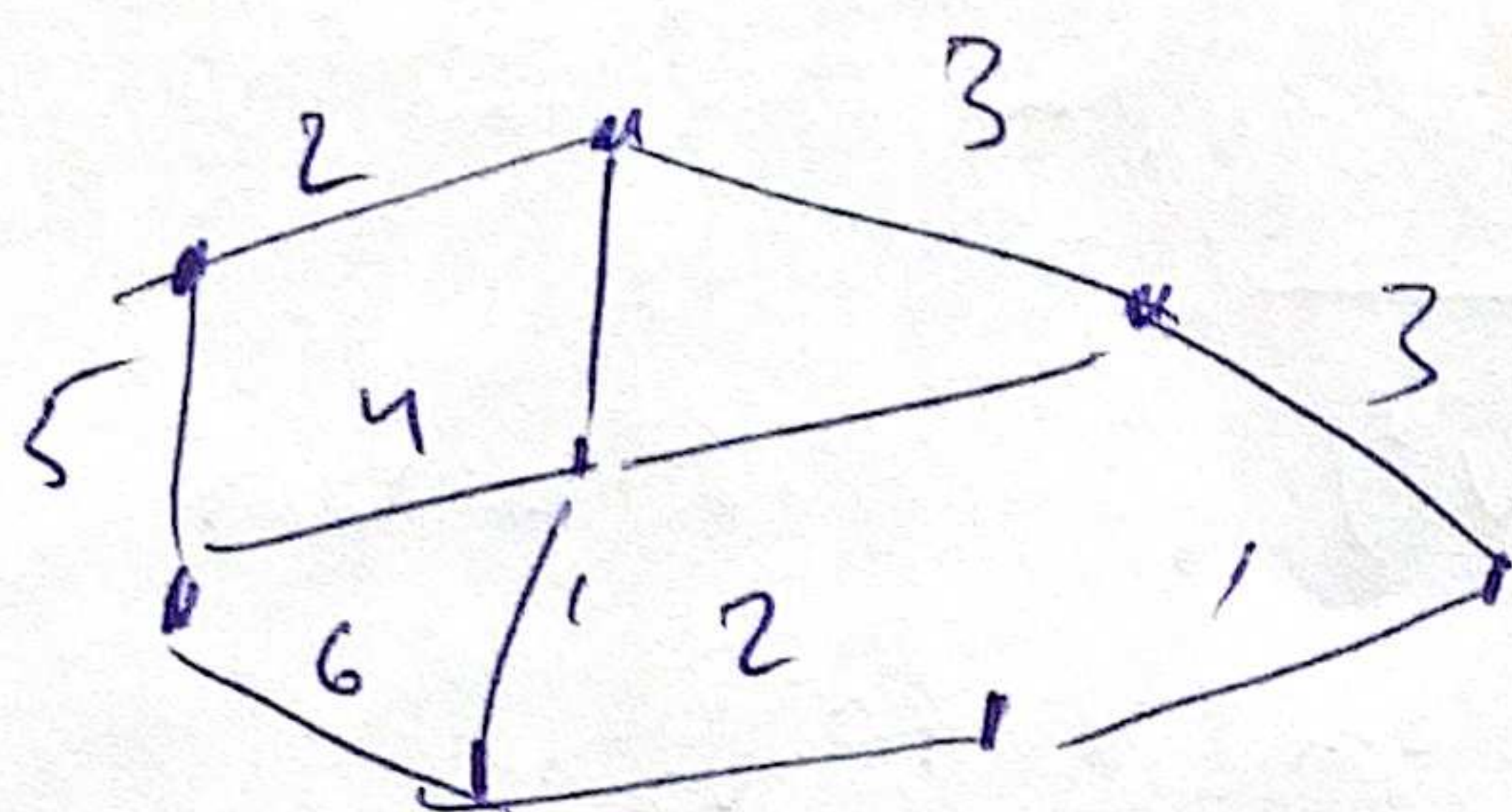
$D_k[i, j] = \min [D_{k-1}[i, j], D_{k-1}[i, k] + D_{k-1}[k, j]]$

$O(V^3)$ runtime, not better than Dijkstra, but a different
 $O(V^2)$ space \rightarrow upto $O(V)$ formulation

Traveling Salesman Problem

NP-complete

vertices = cities



min cost tour:

start at 1, visit all cities and return to 1 at min cost

$(n-1)!$ paths

Stirling for large n

$O(n!)$ in each path compute cost across n vertices

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

with DP we get $O(2^n n^2)$

start at 1, end at 1

to cycle



paths

hard to think recursively

easier to think recursively

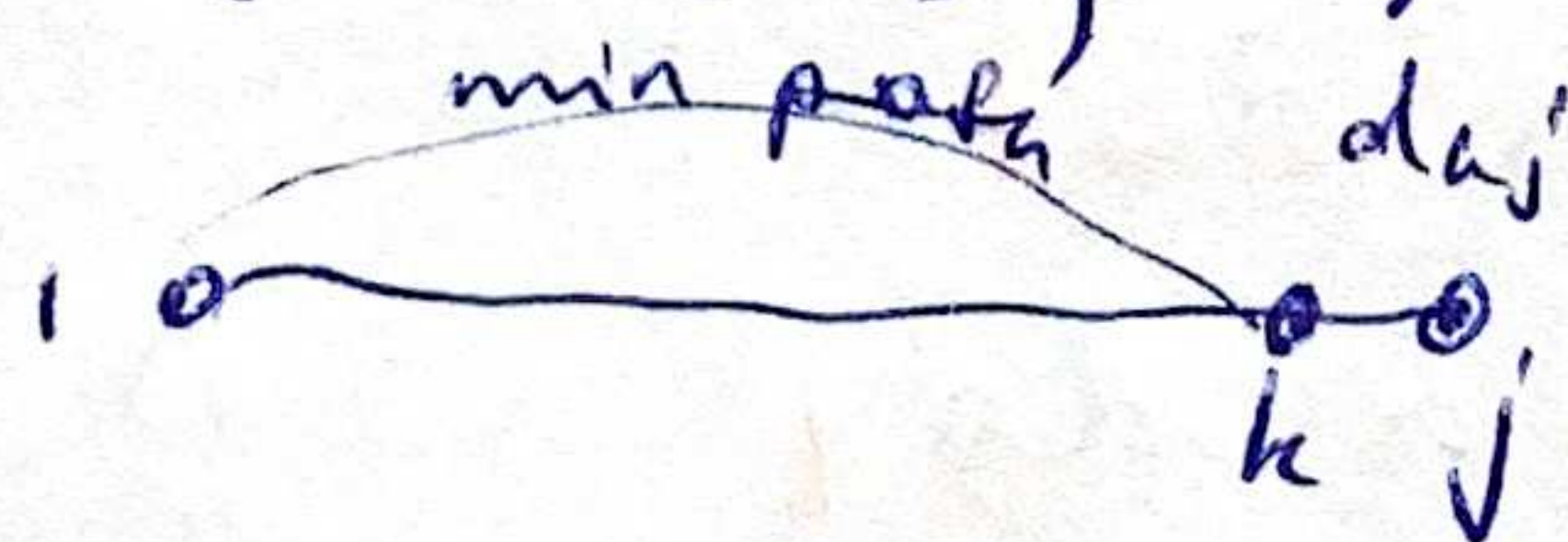
reformulate the problem

start at 1, visit all vertices

$C(S, j)$ = shortest path starting at 1, going through all vertices of S , ending at j

$$C(\emptyset, j) = d_{1j}, \infty \text{ if no edge}$$

$$C(S, j) = \min_{k \in S} [C(S - \{k\}, k) + d_{kj}]$$



which node is k ?
try all of them

for all (S, j) in increasing size of S

$[1, j \notin S]$

$$C(S, j) = \min_{k \in S}$$

precomputed

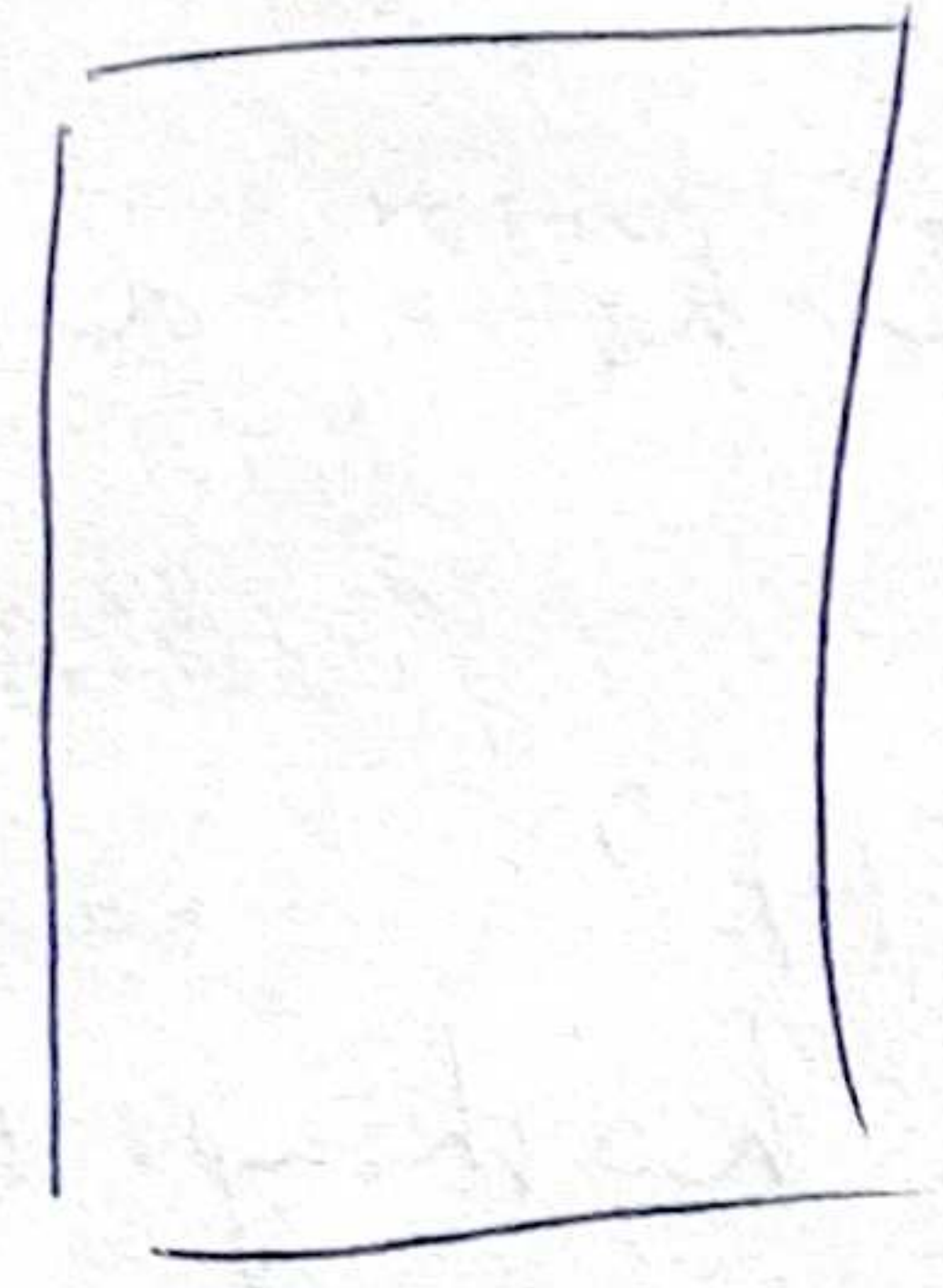
from path to cycle, one more min operation

$$OPT = \min_k [C(\{2, \dots, n\} - \{k\}, k) + d_{k1}]$$

array

vertices j

Subsets
 $S = \{j\}$
but keep
all
subsets of
 S for simplicity



$O(2^n)$ ← memory cells

min op for each cell

$O(n)$

→ $O(2^n n^2)$

↑ runtime

My Summary

DP for strings:

recurrence on string, cost function
↑
reconstruction
↑
edit dist.

DP for graphs:

recurrence on sequence of nodes,
↑
all pairs shortest paths
(subsets of nodes)
↑
TSP

Pattern matching

length

P = pattern

$|P|$

D = document

$|D|$

find pattern in doc.

think as
base 10 numbers

Baseline

char by char move pattern

$O(P \cdot D)$ ← problem for large patterns

hashing

$O(D)$

Hash pattern (eg. 16 bits)

Hash blocks of doc and compare

① why isn't hashing $O(P)$?

↳ can we make hashing $O(1)$?

② False positives?

↳ bound FPs

Pick a prime p . → later discussed, assume as subconstant

hash: number mod p

① work

17935

6386179352...

$$17935 \bmod 251 = 114$$

$$63861 \bmod 251 = 107$$

$$38617 \bmod 251$$

$$p = 251$$

$$N \rightarrow N'$$

$$N' = (N - 10^{|A|-1} \cdot a) \cdot 10 + b$$

$$N' = (N - (10^{|A|-1} \bmod p) \cdot a) \cdot 10 + b \pmod p$$

const precompute

② bound FP rate
pick random prime

$$\pi(x) = \# \text{ of primes } \leq x$$

$$\pi(x) \sim \frac{x}{\ln x}$$

$$10^{100}$$

$$\pi(10^{100}) \sim \frac{10^{100}}{100 \ln 10}$$

$$\text{fraction of primes } \frac{1}{250}$$

const. time
Fingerprinting

Single FP coll
consider (non-padded)
FP coll (is 0 mod p)

$$P - D^* = 0 \bmod p$$

$$P - D^* = 0 \bmod p$$

chunks corresponding to P

$$|P - D^*| \leq 10$$

each size $10^{|A|}$

how many primes can get it to 0 mod p

↳ the number must be product of these primes, at least (each prime can contribute as multiple)

$$j \leq \log_2 X \quad (\text{take } \log_2 \text{ on both sides})$$

Suppose

primes p_1, p_2, \dots, p_j
all divide X , then

$$2^j \leq \prod_{i=1}^j p_i \leq X$$

each prime is at least 2

$$\# \text{ bad Primes} \leq \log_2 X = \log_2 10^{|P|}$$

$$= |P| \log_2 10$$

(3)

of primes that
get a non-matching
pattern to 0 mod p

$$\# \text{ primes (hashing into 64 bits)} = \pi(2^{64})$$

$$P[\text{get a bad prime at a single "non-pattern" hashing}] = \frac{|P| \log_2 10}{\pi(2^{64})}$$

single FP can

$$P[\text{FP in doc}] = \frac{|D| |P| \log_2 10}{\pi(2^{64})}$$

FP across all "non-pattern" hashings in a doc

with k hashes

(increases work by a constant)
but lowers FPrate exponentially

$$P[\text{FP in doc}] = \left[\frac{|D| |P| \log_2 10}{\pi(2^{64})} \right]^k$$