ALGORITHM DESIGN

TL:DR: ONLY WHAT MATTERS, IN DETAIL.

1) KARP-RABIN FINGERPRINT

- HOW TO CHECK EQUALITY OF FILES?

BINARY STRINGS ≡ NUMBERS
S S

DEFINE A FINGERPRINT FUNCTION

$$F(s) = s \bmod p$$

p: RANDOMLY CHOSEN PRIME NUMBER

WE HAVE THE PROBLEM OF THE COLLISION

GIVEN TWO NUMBERS (FILES) k_1, k_2 WE HAVE TWO CASES:

1) $F(k_1) \neq F(k_2) \Rightarrow k_1 \neq k_2$, OK

2) $F(k_1) = F(k_2) \Rightarrow \begin{cases} k_1 = k_2 & \text{OK} \\ k_1 \neq k_2 & \text{FALSE POSITIVE} \end{cases}$

- 1-SIDED ERROR: WHAT IS THE PROBABILITY

THAT THIS HAPPEN?

BEFORE WE COMPUTE THIS PROBABILITY, WE NEED TO:

a) CHOOSE A PARAMETER τ = "SUFFICIENTLY LARGE" — MORE ON THIS LATER

b) CHOOSE A RANDOM PRIME p SO THAT THE ERROR PROBABILITY IS LOW: $p \in [2, \ldots, \tau]$

- LET'S NOW CONSIDER THE FALSE POSITIVE

$$F(k_1) = F(k_2) \wedge k_1 \neq k_2$$

THIS MEANS THAT $k_1 \bmod p = k_2 \bmod p$

OR EQUIVALENTLY $k_1 \equiv k_2 \bmod p$ — CONGRUENCE

WE CALL SUCH p A "BAD PRIME"

WHAT IS THE PROBABILITY THAT WE PICK A BAD PRIME?

$$P(\text{ERROR}) =$$

$$= P_{p \leq \tau}(k_1 \neq k_2 \wedge F(k_1) = F(k_2)) =$$

$$= \frac{\#\text{BAD PRIMES IN } [2, \tau]}{\#\text{PRIMES IN } [2, \tau]} =$$

$$= \frac{?}{\tau / \log \tau} \quad (*)$$

THIS IS A FACT, STRAIGHT FROM THE HEAVENS

1) OBSERVATION: k_1, k_2 REQUIRE N BITS FOR BEING REPRESENTED

2) THEOREM: #DISTINCT PRIME DIVISORS ON ANY NUMBER $< 2^N$ IS AT MOST N

SKIP THE PROOF

1) + 2) $\Rightarrow$ AT MOST #BAD PRIMES $\leq N$

WE CAN CONTINUE FROM $(*)$

$$\leq \frac{N}{\tau / \log(\tau)} = \frac{N}{N^{c+1}} = \frac{1}{N^c} \quad (*)$$

WE KNOW THAT $\frac{\tau}{\log \tau} = N^{c+1}$

WHERE c IS AN ARBITRARY COSTANT

✳: c IS AN ARBITRARY COSTANT

WE CAN CHOOSE τ "SUFFICIENTLY LARGE"

TO MAKE c LARGE AND HAVE A VERY LOW ERROR PROBABILITY

---

2) CUCKOO HASHING

A DICTIONARY IS A DATA STRUCTURE FOR STORING ITEMS THAT SUPPORT 3 OPS.

1) LOOKUP
2) INSERT (IF NOT PRESENT)
3) DELETE (IF PRESENT)

} N.B. THE PRESENTED VERSION DON'T SUPPORT THE DELETION OP.

WITH CUCKOO HASHING: $O(1)$ WORST-CASE (WITH THE PROBLEM OF COLLISIONS)

IDEA:

TAKE TWO HASH FUNCTIONS FROM THE UNIVERSAL FAMILY $\mathcal{H}$

h_1, h_2

WE WANT TO INSERT x:

a) CHECK $T[h_1(x)]$; IF FREE INSERT x

b) IF NOT FREE, CHECK $T[h_2(x)]$ AND IF FREE STORE IT THERE

c) IF NOT FREE: PUT x IN $T[h_1(x)]$, THROWING OUT THE CURRENT ELEMENT y

d) FIND A PLACE FOR y, STARTING WITH $h_2(x)$

TWO OPTIONS:

a) AT SOME POINT YOU FIND A PLACE FOR y

b) IF THERE IS NO PLACE (OR THE OPERATION IS TAKING TOO LONG) WE REBUILD T

- **INSERTION COST**

WE TAKE AT RANDOM h_1, h_2 FROM $\mathcal{H}$.

GIVEN A KEY $x \in U$ AND $i, j \in [M]$, WE HAVE

$$P(h_1(x) = i \wedge h_2(x) = j) \simeq 1/M^2$$

TAKE IT AS A THEOREM: TO REMEMBER IT THINK AS

$$P(h_1(x) = i \wedge h_2(x) = j) = \underbrace{P(h_1(x) = i)}_{1/M} \cdot \underbrace{P(h_2(x) = j)}_{1/M} \Rightarrow 1/M^2$$

NOW CONSIDER THE SET OF KEYS S AND h_1, h_2

S: SET WE WANT TO STORE

WE CAN BUILD AN UNDIRECTED GRAPH $G = (V, E)$ WHERE:

- $V = \{0, 1, \ldots, M-1\}$; THE TABLE POSITIONS
- $E = \{(h_1(x), h_2(x)) : x \in S\}$; THE EDGES ARE RANDOM

INSERTING A NEW ELEMENT x MEANS ADDING AN EDGE $(h_1(x), h_2(x))$ IN G.

AT THIS POINT WE HAVE:

a) EITHER $T[h_1(x)]$ OR $T[h_2(x)]$ IS FREE, $O(1)$

b) WE FOLLOW A PATH IN G UNTIL THE INSERTION COMPLETE

c) AS b) BUT WE TRAVERSE A CYCLE: WE HAVE TO REHASH THE WHOLE TABLE

WE NOW ANALYZE THE CASE b) AND c).

WE ASSUME THAT M (THE SIZE OF THE TABLE) IS $> 2 \cdot c \cdot N$ ($|S| = N$)

COSTANT > 2

b) A FREE POSITION IS EVENTUALLY FOUND

LET i BE THE STARTING POSITION AND j THE FINAL POSITION IN THE PATH.

THE PATH $i \rightsquigarrow j$ HAS LENGTH k IN G WITH THE NOTATION $i \overset{k}{\rightsquigarrow} j$

- LEMMA: $P(\exists\, i \overset{k}{\rightsquigarrow} j) \leq \frac{1}{c^k \cdot M}$

↓

PROOF (BY INDUCTION)

- BASE CASE: $K = 1 \Rightarrow i \overset{1}{\rightsquigarrow} J$ IS AN EDGE

$$P(\exists (i,J) \text{ IN } E) =$$

$$= \sum_{X \in S} P\begin{pmatrix} h_1(X) = i \wedge h_2(Y) = J \\ \vee \\ h_1(X) = J \wedge h_2(Y) = i \end{pmatrix} =$$

$1/M^2$, $1/M^2$ THEOREM

$|S| = N$

$$= N \cdot 2 \cdot 1/M^2 \le \frac{M}{2C} \cdot 2 \cdot 1/M^2 = \frac{1}{C^K M}$$

WITH $K = 1$

WE ASSUMED $M > 2CN \Rightarrow N < M/2C$

- INDUCTIVE CASE: $K > 1$

WE THEN HAVE: $i \overset{K-1}{\rightsquigarrow} R \to J$

PATH EDGE

$$P(\exists i \overset{K}{\rightsquigarrow} J) \le$$

$$\le \sum_{R \in V \setminus \{i,J\}} P(\underbrace{\exists i \overset{K-1}{\rightsquigarrow} R}_{B} \wedge \underbrace{\exists R \to J}_{A}) =$$

$$= P(\exists i \overset{K-1}{\rightsquigarrow} R) \cdot P(\exists R \to J \mid \exists i \overset{K-1}{\rightsquigarrow} R)$$

$P(A \wedge B) = P(A|B) \cdot P(B)$

$P(A|B) \leq 1/c^{k-1}M$ (INDUCTIVE HYP)

$P(B) \leq 1/cM$ (BASE CASE)

$$\leq \frac{1}{c^{k-1}M} \cdot \frac{1}{cM} = \frac{1}{c^k M^2} < \frac{1}{c^k M}$$ ■

NOW, WE KNOW THAT THE INSERTION COST IN THIS CASE IS

$$O(1+k)$$

1: COMPUTE THE 1ST POSITION i

k: FOLLOW THE PATH $i \rightsquigarrow j$

WE WANT TO COMPUTE THE AVERAGE COST!

$$O(1 + E[k]) =$$

$$= O\left(1 + \sum_{k=1}^{N} k \cdot \frac{1}{c^k M}\right) =$$

BY DEF OF EXPECTED VALUE

$$= O(1 + 1/M) = O(1), \text{ OK}$$

$\sum_{K=1}^{M} K/C^K_M = 1/M$ IS A KNOWN SUMMATORY

c) NO FREE POSITION, REHASHING

IN THIS CASE A CYCLE APPEARS AND THUS A REHASHING IS PERFORMED IN $O(N)$ TIME.

WE CHOOSE 2 NEW HASH FUNCTIONS $h'_1, h'_2 \in \mathcal{H}$ AND WE REINSERT EVERYTHING.

WITH WHICH PROBABILITY THIS HAPPENS ?

$$P(\exists \text{ CYCLE IN } G) \leq \quad \text{UNION BOUND: } P(A \cup B) \leq P(A) + P(B)$$

$$\leq \sum_{i=\emptyset}^{M-1} \sum_{K \geq 1} P(\exists\, i \overset{K}{\rightsquigarrow} J, J = i) \leq$$

AS IN CASE b)

$$\leq \sum_{i=\emptyset}^{M-1} \sum_{K \geq 1} 1/C^K_M =$$

$$= \underbrace{\frac{1}{M}}_{\text{TAKEN OUT}} \sum_{i=\emptyset}^{M-1} \sum_{K \geq 1} \underbrace{1/C^K}_{} <$$

$$< \frac{1}{M} \sum_{i=0}^{M-1} \frac{1}{C-1} =$$

$\sum_{k \geq 1} 1/c^k < \frac{1}{C-1}$; KNOWN SUM.

$$= \frac{1}{M} \frac{1}{C-1} \sum_{i=0}^{M-1} 1 = \frac{1}{C-1}$$

WE ASSUMED $C > 2$

$\Rightarrow$ THE REHASHING OCCURS WITH PROB. $p = \frac{1}{C-1} < 1$

AND TAKES $O(n)$

HOW MANY REHASHING?

1 REHASH WITH PROB. p

2 " " " p^2

⋮

t " " " p^t

THE EXPECTED NUMBER OF REHASHING IS

$$\sum_{t \geq 1} t \cdot p^t = O(1)$$

$p < 1$

TO CONCLUDE:

WE ARE INSERTING x THAT WE ASSUME "STARTS" FROM i!

- $P(\exists \text{ CYCLE IN } G, \text{STARTING FROM } i)$

$$= P(\exists\, i \rightsquigarrow j, j = i)$$

$$= \sum_{k \geq 1} \frac{1}{C^k M}$$

$$= \frac{1}{M} \cdot \sum_{k \geq 1} \frac{1}{C^k} \qquad \sum_{k \geq 1} \frac{1}{C^k} < \frac{1}{C-1} \text{ KNOWN}$$

$$< \frac{1}{M} \cdot \frac{1}{C-1}$$

$$= \frac{1}{M} \cdot P \quad \text{AS} \quad P = \frac{1}{C-1}$$

$$= \frac{P}{M}$$

WE PAY THE COST OF REHASHING ($O(N)$) WITH PROBABILITY P/M:

THE EXPECTED COST IS $O(N \cdot P/M) = O(1)$

OF A SINGLE INSERTION

■

3) BLOOM FILTERS

WE HAVE A SET S OF KEYS, $S \subseteq U$.

WE DO NOT WANT TO STORE THEM, AS THEY ARE TOO MANY.

THE GOAL IS TO CHECK $x \stackrel{?}{\in} S$, WITHOUT STORING S.

↓

WE TAKE AT RANDOM k FUNCTIONS $h_1, \dots, h_k \in H$, WHERE $h_i : U \to [M]$.

THEN WE USE A BITVECTOR B OF SIZE M INITIALIZED TO ALL 0s, AND WE DO

$$x \in S \Rightarrow \forall i \in [k].\ B[h_i(x)] = 1$$

THIS TELLS US HOW WE DO THE INSERTION AND THE QUERY $x \stackrel{?}{\in} S$

RETURN $B[h_1(x)] \wedge \dots \wedge B[h_k(x)]$

AS BEFORE: THERE IS THE PROBLEM OF 1-SIDED ERROR

$$x \notin S \wedge \bigwedge_{i \in [k]} B[h_i(x)] = 1$$

LET'S SAY THAT THE PROBABILITY OF ERROR is f:

$$f = P(\text{ERROR}) = P((x \notin S) \wedge (\bigwedge_{i \in [k]} B[h_i(x)]))$$

- TWO POINTS TO CONSIDER:
  1) WHAT is f?
  2) HOW CAN WE CONTAIN f?

1) PROBABILITY ERROR f

SUPPOSE THAT THE BLOOM FILTER B is BUILT, AND TAKE A GENERIC POSITION q AND AN ELEMENT x

STEP a)

$$P(B[q] = 1) \overset{\text{DEF OF PROB.}}{=} 1 - P(B[q] = \emptyset)$$

STEP b)

LETS FIX A FUNCTION h_i.

- $h_i \in H$: $P(h_i(x) = q) = 1/M$
- $B[q] = \emptyset$ WHEN $h_i(x) \neq q$

THEN WE HAVE: $P(h_i(x) \neq q) = 1 - P(h_i(x) = q)$

$$= 1 - 1/M$$

EQUIVALENTLY WE CAN WRITE

$$P(B[q] = \emptyset \mid h_i \text{ IS CHOSEN}) = 1 - \frac{1}{M}$$

STEP c)

LET'S RELAX THE STEP b): DON'T FIX h_i BUT CONSIDER THEM ALL:

$$P(B[q] = \emptyset) = P(\forall i \in [K].\ h_i(x) \neq q) =$$

$$= \left(1 - \frac{1}{M}\right)^K$$

STEP d)

LET'S RELAX THE STEP c): DON'T CONSIDER THE ELEMENT x BUT CONSIDER THEM ALL

$$P(B[q] = \emptyset) = \left(\left(1 - \frac{1}{M}\right)^K\right)^N$$

$P(B[q] = \emptyset)$: PROB $B[q] = \emptyset$ FOR ALL ELEMENTS $x \in S$ AND FOR ALL h_i

$|S| = N$

STEP e)

WE CAN NOW APPLY STEP d) TO THE STEP a)

$$P(B[q]=1) = 1 - P(B[q]=\emptyset)$$

$$= 1 - \underbrace{(1-1/M)^{NK}}_{\text{LET'S CALL IT } p'}$$

$$= 1 - p'$$

WE HAVE THEN COMPUTED f:

$$f = P(\text{ERROR}) = P((x \notin S) \wedge (\bigwedge_{i \in [k]} B[h_i(x)]))$$

$$= (1-p')^{K}$$

K — EVERY POSITION $q \in B$

2) THE BEST PARAMETERS TO MINIMIZE f

WE WANT TO MINIMIZE $f = (1-p')^K$ WHERE $p' = \left(1-\frac{1}{M}\right)^{NK}$

- REMEMBER THE TAYLOR EXPANSION:

$$(1+\gamma) \simeq e^{\gamma} \text{ FOR SMALL } \gamma$$

$$p' = \underbrace{\left(1-\frac{1}{M}\right)}_{\gamma}^{NK} \simeq \left(e^{-1/M}\right)^{NK} = \underbrace{e^{\frac{-NK}{M}}}_{\text{LET'S CALL THIS } p}$$

AND $p' \simeq p$

WE NOW TRY TO MINIMIZE $(1-p)^k$

LETS EXPLOIT THE LOGS:

$$p = e^{\frac{-Nk}{M}} \Rightarrow \log(p) = \log_e\left(e^{-Nk/M}\right) =$$

PROP. LOGS

$$= -\frac{Nk}{M}\log(e)$$

$\log_e(e) = 1$

$$= -\frac{Nk}{M}$$

$$\log(p) = -\frac{Nk}{M} \Rightarrow k = -\frac{M}{N}\log(p)$$

WE HAVE FOUND K!

- TAKE LOGS:

$$f \simeq (1-p)^k \Rightarrow \log(f) = \log\left((1-p)^k\right) =$$

NOT p'

$$= k\log(1-p) =$$

$$= -\frac{M}{N}\log(p)\log(1-p)$$

k

$$\log(f) = -\frac{M}{N}\log(p)\log(1-p)$$

↓

THIS TELLS US "HOW MINIMIZE" STUFF:

WE MINIMIZE OVER p, AND THE BEST CHOICE FOR p is $1/2$; THEN:

$$f \simeq (1-p)^k = \left(1-\frac{1}{2}\right)^k = \left(\frac{1}{2}\right)^k = \frac{1}{2^k} =$$

$$= \frac{1}{2^{-M/N\log(1/2)}} = \boxed{\frac{1}{2^{M/N\log(2)}}} =$$

SEE LATER ✱

$\log(a/b) = \log(a) - \log(b)$

$\log(1/2) = -1$

$$= \left(2^{-\log 2}\right)^{M/N} \simeq 0.618^{M/N}$$

$= 0.618$

(YOU HAVE TO CHOOSE M ACCORDINGLY)

A PRETTY LOW ERROR PROBABILITY

$\log(1 - \log(2))$

↓

SPACE COMPLEXITY OF BF

THE SPACE COMPLEXITY IS

$$M + \underbrace{O(k)}_{\text{WORDS}}$$

BITS TO STORE THE BITVECTOR

STORE k HASH FUNCTIONS: STORING A PAIR (a,b) FOR EACH h_i

CONSIDER f:

* $f = \frac{1}{2^{M/N \log(2)}} \Rightarrow \log(f) = \frac{M}{N}\log(2)\cdot\underbrace{\log(\tfrac{1}{2})}_{=-1} \Rightarrow$

$$\Rightarrow M \simeq \frac{N\log(f)}{\log(\frac{1}{2})\log(2)} = \frac{N\log(1/f)}{\log(2)} = 1.44 \cdot N\log(1/f) \text{ BITS}$$

---

4) LOAD BALANCING

M SERVERS, N JOBS

- FAIR LOAD: M/N JOBS PER SERVER

RANDOMLY CHOOSE AN HASH FUNCTION $h \in H$ USING h SO THAT WE HAVE JOB i ASSIGNED TO THE MACHINE $h(i)$

↳ THIS GIVES A FAIR LOAD

PROOF:

X_J = LOAD FOR THE MACHINE J = #JOBS ASSIGNED TO J

LET'S DEFINE X_{ij}:

$$X_{ji} = \begin{cases} 1 & \text{IF THE JOB } i \text{ IS ASSIGNED TO } \underbrace{J = h(i)}_{\text{WITH PROB } P = 1/M} \\ \phi & \text{OTHERWISE} \end{cases}$$

SO WE CAN COMPUTE

- $X_J = \sum_{i=\phi}^{N-1} X_{ji}$
- $E[X_J] = \sum_{i=\phi}^{N-1} E[X_{ji}] = \sum_{i=\phi}^{N-1} P(X_{ji} = 1) = \frac{N}{M}$ ∎

($\sum_{i=\phi}^{N-1}$: N terms; $P(X_{ji} = 1)$: $1/M$)

$E[X_J] = E[\sum_{i=\phi}^{N-1} X_{ji}]$ — DEF OF X_J

$= \sum_{i=\phi}^{N-1} E[X_{ji}]$ — LINEARITY OF E

WHAT ABOUT THE MAX LOAD?

TO COMPUTE THE MAX LOAD WE FIRST NEED TO REMEMBER TWO RULES:

a) $P(X_{ji} = 1 \wedge X_{ji'} = 1) = P(X_{ji}=1)P(X_{ji'}=1)$

$= P(h(i)=j)P(h(i')=j)$

TWO DIFFERENT TASKS i, i' BOTH ASSIGNED TO THE SAME MACHINE j

AS $h \in \mathcal{H}$ IS 2-WAY INDEPENDENT!

b) X, Y, Z PAIRWISE INDEPENDENT

INDICATOR VARIABLES

$\Rightarrow \sigma^2(X+Y+Z)$

$=$

$\sigma^2(X) + \sigma^2(Y) + \sigma^2(Z)$

VARIANCE:

$VAR(X) = \sigma^2(X) = E[X^2] - E^2[X]$

HOW FAR FROM THE MEAN A R.V'S VALUES ARE LIKELY TO BE.

FROM a) AND b) WE CAN SAY THAT

$$\sigma^2(X_J) = \sum_{i=\emptyset}^{N-1} \sigma^2(X_{Ji})$$

a) TELLS US THAT OUR RANDOM VARS ARE PAIRWISE INDEPENDENT!

THE FACT THAT $X_{Ji} = 1$ TELLS US NOTHING ABOUT THE VALUE OF $X_{Ji'}$. THIS FOR EACH i,J POSSIBLE

HENCE WE CAN USE b) WITH OUR RVs.

THEN, WE KNOW THAT

$$\sigma^2(X_J) = \sum_{i=\emptyset}^{N-1} \sigma^2(X_{Ji}) =$$

$$= \sum_{i=\emptyset}^{N-1} \left(E[X_{Ji}^2] - E^2[X_{Ji}] \right) =$$

DEF OF σ^2

$E[X_{Ji}] = 1/M$: $X_{Ji}^2 = \begin{cases} 1^2 \dots \\ \emptyset^2 \dots \end{cases}$

$\left(\frac{1}{M}\right)^2 = \frac{1}{M^2}$

$$= N\left(\frac{1}{M} - \frac{1}{M^2}\right)$$

HENCE WE HAVE

$$\sigma = \sqrt{N\left(\frac{1}{M} - \frac{1}{M^2}\right)}$$

NOW, TO COMPUTE (OR BETTER SAID, BOUND) THE MAX LOAD OF A MACHINE WE USE THE

CHEBYCHEV'S INEQUALITY

$$P(|X - E[X]| \geq k\sigma) \leq \frac{1}{k^2}$$

- $|X - E[X]|$: DISTANCE FROM X AND ITS EXP. VALUE
- $k\sigma$: A CONSTANT OF OUR CHOOSING

NOW WE SET $k = \sqrt{2M}$.

THEN $\sigma k = \sqrt{2M} \cdot \sqrt{N\left(\frac{1}{M} - \frac{1}{M^2}\right)} = \sqrt{2 \cdot N\left(1 - \frac{1}{M}\right)} \cong \sqrt{2N(1)} = \sqrt{2N}$

- $\sqrt{a} \cdot \sqrt{b} = \sqrt{ab}$
- k
- σ
- ~ 0 AS $M \gg 0$
- NUMBER OF SERVERS

AND WE GET

$$P\left(|X_j - \frac{N}{M}| \geq \sqrt{2N}\right) \leq \frac{1}{2M}$$

- $E[X_j]$
- σk
- k^2

WE CAN THEN COMPUTE THE FOLLOWING PROBABILITY

$$P\left(\max_j : \left|X_j - \frac{N}{M}\right| \leq \sqrt{2N}\right) =$$

$$= 1 - P\left(\exists j : \left|X_j - \frac{N}{M}\right| \geq \sqrt{2N}\right) \quad a)$$

$$\geq 1 - \sum_{j=0}^{M-1} P\left(\left|X_j - \frac{N}{M}\right| \geq \sqrt{2N}\right) \quad b)$$

UNION BOUND
a) ≤ b)

$$= 1 - \sum_{j=0}^{M-1} \frac{1}{2M}$$ / CHEBYSHEV'S

$$= 1 - \frac{1}{2}$$ / — SIMPLIFY THE $\sum_{j=0}^{M-1} 1 = M$

$$\geq \frac{1}{2}$$

HENCE:

$$P\left(\max_j : \left|X_j - \frac{N}{M}\right| \leq \sqrt{2N}\right) \geq \frac{1}{2}$$

THE MAX LOAD (THE LOAD OF THE MOST LOADED MACHINE j) IS AT MOST $N/M + \sqrt{2N}$, WITH PROB. $\geq \frac{1}{2}$

GREATER THAN 1/2

---

5) COUNT-MIN SKETCHES FOR FREQ. ELTS

CONSIDER A STREAM (POSSIBLY INFINITE) $a_0 \ldots$ OF ELEMENTS $\in U$

WE WANT TO KNOW HOW MANY TIMES A GIVEN ELEMENT a HAS APPEARED SO FAR IN THE STREAM.

LET'S DEFINE A FREQUENCY ARRAY:

$$F[a] = \#\text{TIMES } a \text{ APPEARED SO FAR}$$

THE POINT IS THAT $\|F\| = \sum_{a \in U} F[a]$ IS NOT (↑ / a ∈ U)

NORM

COMPUTABLE WHEN U IS HUGE (OR WHEN THE STREAM IS INFINITE)

HUGE → NOT ENOUGH MEMORY

FOR SOLVING THIS PROBLEM WE FIRST SET 2 OBJECTIVES:

1) SMALL SPACE OCCUPATION

2) 2 USER DEFINED PARAMETERS:

- δ: THE ERROR PROBABILITY
- ε: WE CAN'T COMPUTE $F[\cdot]$ EXACTLY, SO WE COMPUTE AN ε-APPROX

THE ALGORITHM THAT SOLVE THE PROBLEM IS CALLED COUNT-MIN SKETCH.

WE SET: (KINDA THE ALPHABET OF THE STREAM)

- $|U| = N$
- UPDATE OP. FOR F: $F[i]$++ FOR EVERY $i \in [N]$
- QUERY OP.: RETURN $F[i]$
- $\|F\| = \sum_{a \in U} F[a]$ } DEF

AND GIVEN ε, δ WE COMPUTE AN APPROXIMATION $\tilde{F}[i]$ WHERE

$$F[i] \leq \tilde{F}[i] \leq F[i] + \varepsilon \|F\|$$

THIS IS TRUE WITH PROB. $1 - \delta$

THE PROBLEM IS THAT IF $F[i]$ IS SMALL COMPARED TO $\varepsilon \|F\|$ THE ESTIMATION IS USELESS

- INSIGHT: CM-SKETCH IS A SORT OF BLOOM FILTERS WITH COUNTERS.

GIVEN THE INPUT ε AND δ WE BUILD A TABLE $T = R \times C$, WHERE

- R : ROWS, $R = \log\left(\frac{1}{\delta}\right)$
- C : COLUMNS, $C = \frac{e}{\varepsilon}$

AND EACH ENTRY OF THE TABLE IS A COUNTER INITIALIZED TO $\emptyset$.

WE THEN CHOOSE UNIFORMLY AND RANDOMLY $h_0 \dots h_{R-1} \in \mathcal{H}$ HASH FUNCTIONS WITH $\mathcal{H}$ A TWO-WAY INDEPENDENT UNIVERSAL HASH FUNCTION, WHERE $M = C$.

→ SEEN BEFORE IN LOAD BALANCING

THE ITEM $i \in U$ OF THE STREAM IS ASSOCIATED WITH

THE ENTRIES AT POSITIONS

$$\{\langle i, h_0(i)\rangle \dots \langle i, h_{R-1}(i)\rangle\}$$

WE THEN HAVE THE TWO OPERATIONS:

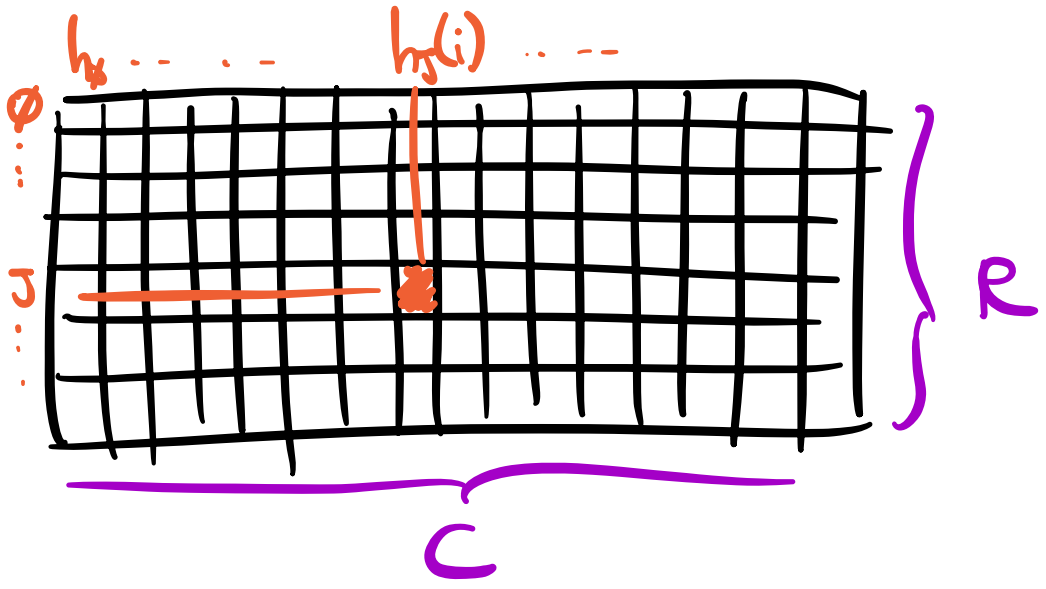
- UPDATE: A NEW ELEMENT i IS SEEN IN THE STREAM:

  $F[i]{+}{+} \equiv$ INCREMENT BY 1 THE CELLS AT POSITIONS $\langle j, h_j(i)\rangle$ FOR $j \in [0, R-1]$

- QUERY:

  RETURN $\min_{0 \le j \le R-1} T[j][h_j(i)]$ ✱

$T =$



IT'S CLEAR THAT THE OPERATIONS COST $O(R)$

⊛ ISSUE: $\tilde{F}[i] > F[i]$ BY CONSTRUCTION, BUT IT COULD BE TOO LARGE!

LET j BE THE MIN OF ⊛

THEN

$$\tilde{F}[i] = T[j][h_j(i)] = F[i] + X_{ji}$$

WHERE X_{ji} IS "GARBAGE" DUE TO THE UPDATES $F[k]$ ++ OF ELEMENTS $k \neq i$ (COLLISIONS!)

X_{ji} IS INCREASING (W.R.T $F[i]$) THAT THE CELL $T[j][h_j(i)]$ HAD DUE TO THE COLLISIONS OF ELEMENTS $k \neq i$.

IF ↑ THIS COLLISION HAPPENS ($h_j(i) = h_j(k), i \neq k$), IT HAPPENS N_k THE TIMES WE SEE k IN THE STREAM, HENCE $F[k]$ TIMES.

MORE SPECIFICALLY:

$$X_{ji} = \sum_{\substack{k \neq i \\ h_j(i) = h_j(k)}} F[k] = \sum_{k=1}^{N} I_{jik} \cdot F[k]$$

WHERE I_{Jik} IS AN INDICATOR VARIABLE:

$$I_{Jik} = \begin{cases} 1 & \text{IF } h_j(i) = h_j(k), \text{ WITH } i \neq k \\ 0 & \text{OTHERWISE} \end{cases}$$

WE WANT TO SHOW: $P(X_{Ji} > \varepsilon \|F\|) < \delta$ ★ AT THE END

↓

WE COMPUTE

DEF OF IND. VAR

$$E[I_{Jik}] = P(I_{Jik} = 1) = \frac{1}{c} = \frac{\varepsilon}{e}$$

$c = \frac{e}{\varepsilon}$

$h_j : U \to [c]$ $\mathcal{H}$ IS UNIVERSAL: THE PROB A COLLISION IS AT MOST $\frac{1}{c}$

WE USE IT TO COMPUTE THE EXPECTED VALUE OF X_{Ji}:

$$E[X_{Ji}] =$$

DEF OF X_{Ji} + LINEARITY OF E

$$= \sum_{k=1}^{N} E[I_{Jik}] \cdot F[k]$$

JUST COMPUTED

$$= \sum_{k=1}^{N} \frac{\varepsilon}{e} \cdot F[k]$$

TAKEN OUT NOT APPEAR K

$$= \frac{\varepsilon}{e} \sum_{k=1}^{N} F[k]$$

DEF OF $\|F\|$

$$= \frac{\varepsilon}{e} \|F\|$$

WE SHOWED

$$E[X_{ji}] = \frac{\varepsilon}{e}\|F\| \Rightarrow eE[X_{ji}] = \varepsilon\|F\|$$

WE CAN APPLY THE MARKOV'S INEQUALITY:

$$P(X > z) \leq \frac{E[X]}{z}$$

IN THIS CASE WE SET

- $z = \varepsilon\|F\|$
- $X = X_{ji}$

AND WE GET: M.I

$$P(X_{ji} > \varepsilon\|F\|) \overset{1}{\leq} \frac{E[X_{ji}]}{\varepsilon\|F\|} \overset{*}{=} \frac{E[X_{ji}]}{eE[X_{ji}]} = \frac{1}{e}$$

AND WE CAN FINALLY COMPUTE:

$$P(\tilde{F}[i] > F[i] + \epsilon \|F\|) =$$

$$= P(\forall j : P(X_{ji} > \epsilon \|F\|))$$

$\tilde{F}[i]$ IS A MINIMUM OPERATION OVER THE ROWS OF T: IT HAS TO BE VERIFIED FOR EVERY CELL!

$$\leq \prod_{j=1}^{R} \frac{1}{e}, \text{ As } h_j \in \mathcal{H}$$

← COMPUTED BEFORE ↑

$$= \frac{1}{e^R}$$

$$= \delta$$

$$R = \log(1/\delta)$$

$$\Rightarrow$$

$$\frac{1}{e^{\log(1/\delta)}} \equiv \frac{1}{\frac{1}{\delta}} = \delta$$

WHICH PROVES ★ ■

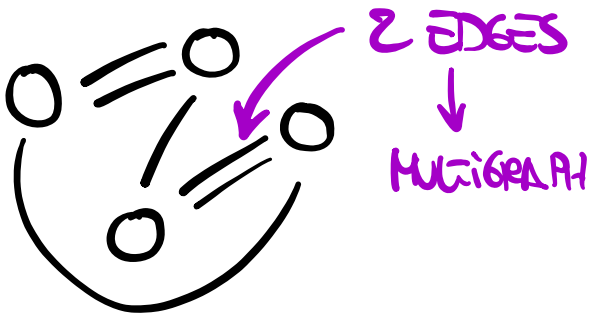
6) (RANDOMIZED) MIN-CUT ALGORITHM

CONSIDER AN UNDIRECTED (MULTI) GRAPH

$G=(V,E)$, WHERE

- $|V| = N$
- $|E| = M$

MULTIPLE EDGES BETWEEN THE SAME NODES



WE DEFINE A CUT OF THE GRAPH

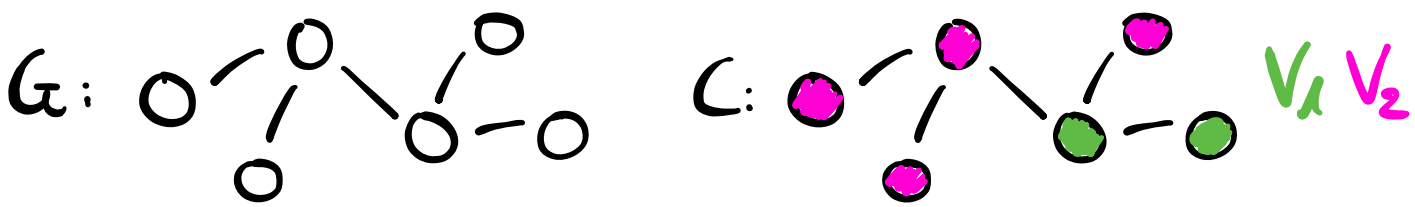
$$C = (V_1, V_2)$$

WHERE

- $V_1 \cap V_2 = \emptyset$
- $V_1 \cup V_2 = V$

} WE CALL V_1, V_2 PARTITIONS OF V

PARTITIONS CAN BE SEEN AS COLORINS



WE THEN DEFINE A CUT-SET OF G

$$E(V_1, V_2) = \left\{ uv \in \bar{E} : \begin{array}{c} (u \in V_1 \wedge v \in V_2) \\ \vee \\ (u \in V_2 \wedge v \in V_1) \end{array} \right\}$$

SAID EASY: SET OF EDGES THAT CROSS THE PARTITIONS OF A CUT.
ALSO NOTATED WITH $E(C)$

THE PREVIOUS DEFINITIONS ARE USED TO DEFINE SUBGRAPHS OF G:

- NODE INDUCED (SUB) GRAPHS:

$$G[V_i] = G(V_i, \bar{E}_i) = G(u, v \in V_i, uv \in \bar{E})$$

- SUBGRAPH INDUCED BY THE CUT-SET

$$G[E(V_1, V_2)] = (V_{1,2}, \bar{E}(V_1, V_2))$$

$$\text{WITH } V_{1,2} = \{ u \in V : uv \in \bar{E}(V_1, V_2) \}$$

"NODES OF THE CUTSET"

COMPUTING THE SIZE OF $G[E(C)]$ IS HARD

- COMPUTING THE LOWER BOUND TAKE POLY TIME

- COMPUTING THE UPP. BOUND IS NP-HARD

WE SAY THAT A CUT $C = (V_1, V_2)$ IS A MIN-CUT IF IS TRUE THAT

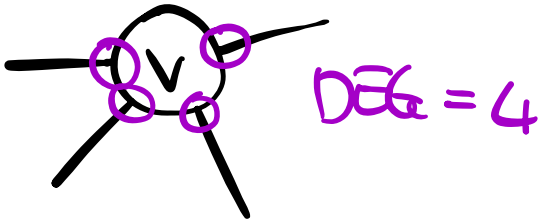
$$\forall C'. |E(C)| \leq |E(C')|$$

THE CUT C IS A MIN-CUT IF THE ASSOCIATED CUT-SET IS THE SMALLEST CUT-SET.
THE CUT WITH THE LEAST AMOUNT OF EDGES

- OBSERVATION: THE MIN-CUT IS NOT UNIQUE
- OBSERVATION: $\forall v \in V. |E(C)| \leq DEG(v)$

WITH C MIN-CUT

DEGREE OF V:
EDGES OF V



EDGE CONTRACTION

CONSIDER AN EDGE $uv \in E$

WE DEFINE THE GRAPH $G / \{uv\} = (V', E')$, WHERE

- $V' = (V - \{u, v\}) \cup \{\overline{uv}\}$

- $E' = (E - \{uv\}) \cup \{\overline{uv}z : uz \in E \vee vz \in E\}$

CONTRACTING AN EDGE UV MEANS TO "MERGE" THE NODES U AND V IN A SINGLE NODE UV AND THEN SUBSTITUTE ALL THE EDGES WHERE EITHER U OR V APPEAR WITH EDGES TO/FROM UV

- INTUITION: WHEN PERFORMING THE EDGE CONTRACTION WITH $u, v \in V$, WE ARE SAYING THAT u, v BELONG TO THE SAME "SIDE" OF THE CUT.

CONSIDER A CONNECTED GRAPH G:

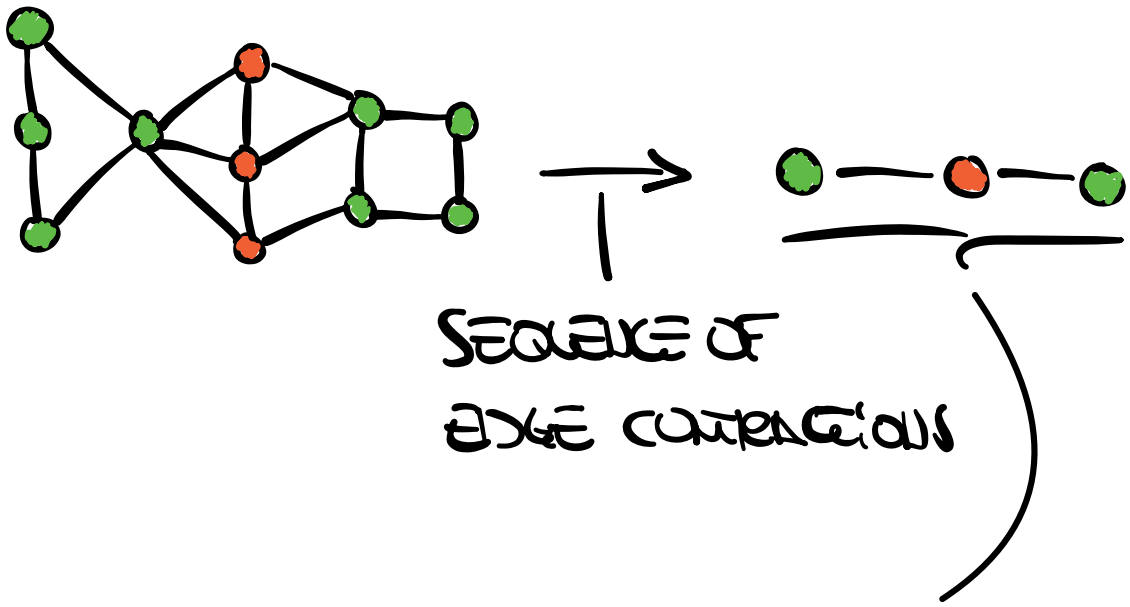
EVERY TWO NODES ARE CONNECTED BY AN EDGE

THEOREM A)
ANY SEQUENCE OF EDGE CONTRACTION LEAVING JUST 2 NODES GIVES A CUT $C(V_1, V_2)$

PROOF BY INDUCTION ON #NODES

THEOREM B)
GIVEN A CUT $C = (V_1, V_2)$, IT DOES NOT NECESSARILY MEANS THAT EXISTS A SEQUENCE OF EDGE CONTRACTION THAT LEADS TO JUST 2 NODES

PROOF)



THE CUT 🟥 🟩 IS NOT OBTAINABLE
BY EDGE CONTRACTION

WE HAVE NOW ALL THE INGREDIENTS TO DEFINE AN ALGORITHM THAT "GUESSES" THE MIN-CUT OF G

THE SIZE!

```
GUESSMINCUT(G=(V,E)){
  WHILE(|V|>2)
    {1) CHOOSE RANDOMLY UV ∈ E
     2) G = G - UV } EDGE CONTRACTION
  RETURN |E| } NUMBER OF EDGES THAT REMAIN WITH JUST 2 NODES (MULTIGRAPH!
}
```

HOW WE CHOOSE AN EDGE $UV \in E$?

- CHOOSE U WITH PROB. $1/|V|$

- CHOOSE $v \in V(u)$ WITH PROB $\frac{1}{DEG(u)}$

NOTATION: SET OF VERTEXES CONNECTED TO u

THANKS TO THEOREM A) THE ALGORITHM CAN PROVIDE THE SIZE OF A MIN-CUT

WHEN THE PREVIOUS ALGORITHM FAIL?

IT FAILS WHEN AN EDGE uv IS A BAD EDGE, WHICH IS AN EDGE THAT BELONGS TO EVERY POSSIBLE MIN-CUT.

IN SYMBOLS:

$uv \in E$ IS BAD WHEN $\forall$ MIN-CUT C. $uv \in E(C)$

uv BELONGS TO EVERY MIN-CUT SETS!

IN WORDS: GUESSMINCUT CAN'T FIND A MIN-CUT WHEN IT REMOVES AN EDGE THAT WOULD BELONG TO EVERY MIN-CUT. AKA EDGE CONTRACTION

LET $k = |E(C)|$ BE THE SIZE OF A MIN-CUT SET.

PROPOSITION:

THERE ARE AT MOST K BAD EDGES

THE PROOF IS THE INTUITIVE ONE

SO WE HAVE THAT

$$P(uv \text{ IS BAD}) = \frac{k}{m}$$

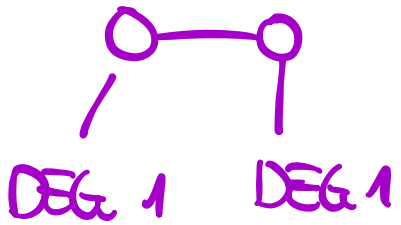
$|E| = m$

(*) SEE LATER

WE KNOW THAT

1) $\sum_{u \in V} DEG(u) = 2m$

UNDIRECTED GRAPH EVERY EDGE COUNTED TWICE



2) $\forall u \in V : DEG(u) \geq k$

OTHERWISE THE MIN-CUT SIZE WOULD BE SMALLER, BUT K IS THE SIZE OF $\bar{E}(C)$

DUMB INSIGHT → IN A GUESSMINCUT CONTEXT: ONLY TWO NODES:

↳ THIS IS K!

1) ∧ 2) $\Rightarrow$ $2M > NK$ $\Leftrightarrow$ $M \geq \frac{NK}{2}$

2) 1) |V| 2)

WE CAN USE IT IN ✱ (SEE ABOVE)

$$P(uv \text{ IS BAD}) \leq \frac{K}{M} \leq \frac{K}{\frac{NK}{2}} = \frac{2}{NK} \cdot K = \frac{2}{N}$$

WE THEN HAVE

- $P(N)$ = GUESSCUTMIN FINDS $|ECC)|$ FOR THE GRAPH $G=(V,E)$, $|V|=N$
- $P(N) \geq \begin{cases} 1 \text{ WITH } N=2 \\ \left(1-\frac{2}{N}\right) P(N-1), \text{ WITH } N>2 \end{cases}$

PROB. THE CHOSEN EDGE IS NOT A BAD EGE

GUESS MINCUT WITHOUT THE MERGED NODE

WE EXPAND THAT RECURSIVELY

- NOTE: $1-\frac{2}{N} = \frac{N-2}{N}$

$$P(N) \geq \left(\frac{N-2}{N}\right) \cdot \left(\frac{N-3}{N-1}\right) \cdots \frac{1}{3} \cdot 1 = \frac{(N-2)!}{\frac{N!}{2}} = \frac{2}{N(N-1)} \blacksquare$$

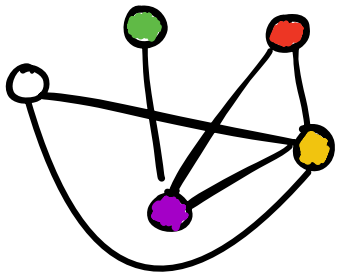
---

7) VERTEX COVER

WE NOW CONSIDER THE VERTEX COVER PROBLEM, WHICH IS NP-COMPLETE.

GIVEN AN UNDIRECTED GRAPH $G = (V, E)$, WE SAY THAT:

$S \subseteq V$ IS A VERTEX COVER IF $\forall uv \in E : (u \in S) \vee (v \in S)$



$S = \{ \text{●}, \text{●} \}$

VERTEX COVER

WE CAN NOW DISTINGUISH BETWEEN THE VERTEX COVER PROBLEM AND ITS DECISIONAL FORM

1) MINIMIZATION: FIND $K_{MIN} : |S| = K_{MIN}$ S.T S IS A VERTEX COVER OF MINIMAL SIZE:

$$\forall S'. |S'| < |S| \Rightarrow S' \text{ IS NOT A V.C.}$$

2) DECISION: CONSIDER VC_k, A VERTEX COVER OF SIZE K, DOES EXISTS A SMALLER VC?

↳ AKA: IS VC_k MINIMAL?

WE FOCUS ON DECISION

- BASELINE FOR VERTEX COVER

THERE ARE $\binom{N}{K} \sim N^K$ SUBSETS S OF V

WE CAN CHECK EACH SUBSET IN POLYTIME,

HENCE THE COST IS $O(N^K \cdot POLY(N))$ $(O(N^{K+O(1)}))$??

- N^K: # SUBSETS
- $POLY(N)$: PRICE FOR EACH SUBSET
- $N^{K+O(1)}$: GROSSI NOTES

PARAMETRIZED ALGORITHMS: { EXAG EXPONENTIAL ALGORITHMS

WE CAN SOLVE THE PROBLEM IN EXACTLY $O(f(k)\,POLY(N))$ TIME, WHERE $f(k)$ CAN BE LARGE.

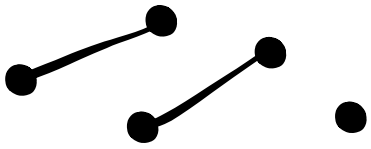
THIS IS USEFUL WHEN $K \ll N$.

A) KERNELIZATION

CONSIDER AN UNDIRECTED GRAPH $G = (V, E)$.

LET'S CONSIDER THE EASY CASES:

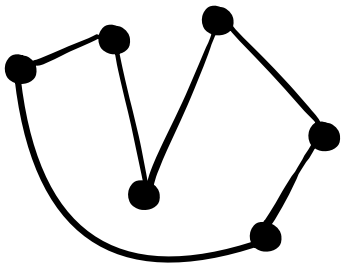
CASE 1:

$G =$ 

THE MAXIMUM DEGREE OF A NODE IS 1:

$$MAX\_DEGREE(G) = 1$$

CASE 2:

$G =$ 

IN THIS CASE

$$MAX\_DEGREE(G) = 2$$

CONSIDER $G = (V, E)$. WITH $\langle G, k \rangle$ WE MEAN THAT WE WANT TO CHECK WETHER A VERTEX COVER OF SIZE $\leq k$ DOES EXISTS IN G.

WE THEN APPLY THE FOLLOWING TWO RULES AS LONG AS POSSIBLE:

R1) IF v IS AN ISOLATED NODE ($DEG(v) = \emptyset$) THEN
SOLVE $\langle G - \{v\}, k \rangle$

R2) IF $DEG(v) > k$ THEN SOLVE $\langle G - v, k-1 \rangle$

IT'S CLEAR THAT R1) AND R2) TAKES $POLY(n)$ TIME.
WHEN R1) AND R2) ARE NOT APPLICABLE ANYMORE WE APPLY THE RULE

R3) • IF $(k < \emptyset) \vee (G$ HAS $> k + k^2$ NODES$) \vee$
$(G > k^2$ EDGES$)$

• THEN:
REPORT NO VERTEX COVER

• ELSE:
G_K BE THE CURRENT GRAPH (G REDUCED WITH R1 AND R2), SOLVE VERTEX COVER ON $\langle G_K, k \rangle$ AND RETURN THE ANSWER

THE REDUCED GRAPH WITH R1) AND R2) IS SAID THE KERNEL OF G.
THAT'S WHY KERNELIZATION.
└ UNDR. MY SPECULATIONS

R3)

IF $\langle G_{K,1}, K\rangle$ HAS A V.C. OF SIZE $\leq K$ THEN G_K MUST HAVE $\leq K^2 + K$ NODES (2) AND $\leq K^2$ EDGES (1)

LET S BE SUCH V.C.

1)
- $|S| \leq K$ — DEF OF $\langle G, K\rangle$
- $\forall u \in S.\ \text{DEG}(u) \leq K$ $\Rightarrow |E| \leq |S| \cdot K = K^2$
- S IS A VC — DEF OF $\langle G, K\rangle$

PERCHÉ? TODO

2) $|S| + |G_K - S| \leq K + K^2$

($|S| \leq K$)

THE APPLICATION OF THE RULES TAKE POLY TIME, SO:

1) WE TAKE A GRAPH
2) APPLY THE RULES AND OBTAIN $\langle G_{K'}, K\rangle$
3) RUN THE BASELINE ON $\langle G_K, K\rangle$

$$O((K^2)^K + \text{POLY}(N)) = O(f(K) + \text{POLY}(N)) \blacksquare$$

$(K^2)^K$: FUNCTION IN K, BASELINE COST

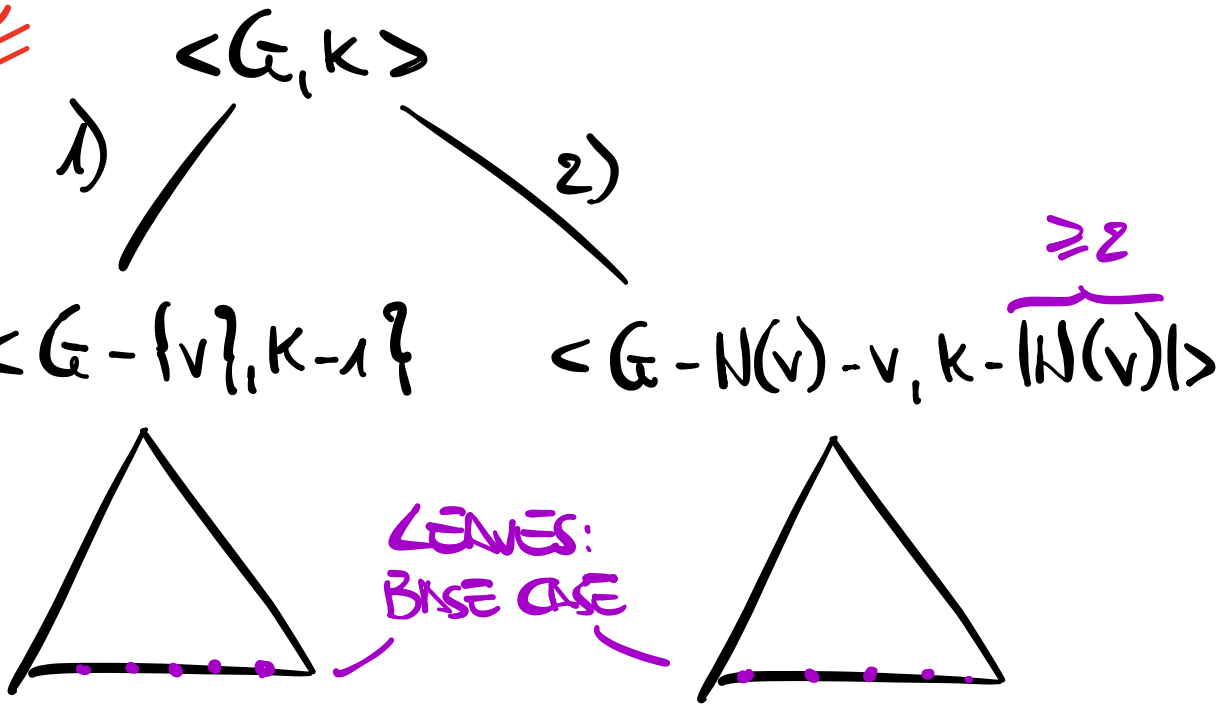
B) BRANCHING TECHNIQUE

LET'S CONSIDER $\langle G, K \rangle$ AS BEFORE

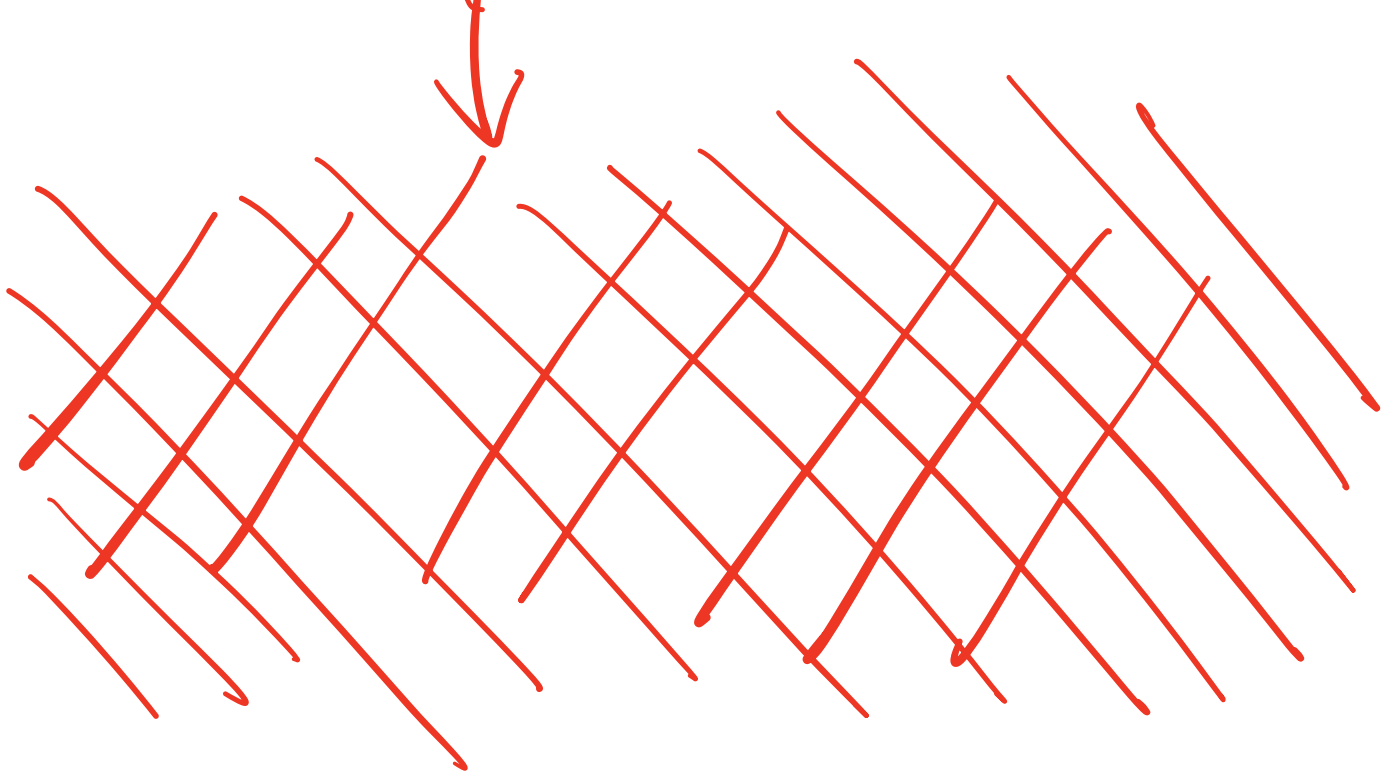
LET'S DEFINE V AS THE NODE IN G WITH MAXIMUM DEGREE, WE WILL HAVE RECURSION ON THIS DEGREE.

- BASE CASE: $DEG(V) = 1$
- RECURSIVE CASE: $DEG(V) > 1$
  1) EITHER $V \in VC$, AND WE CAN'T SAY ANYTHING FOR THE NODES $\in N(V)$, OR
  2) EVERY NODE IN $N(V)$ BELONG TO VC

COME



TODO: COME FUNZIONA?



THE TOTAL COST IS GIVEN BY

$$\#\text{RECURSIVE CALLS} \cdot \text{POLY}(N)$$

EACH CALL GENERATES TWO CALLS, HENCE

$$\#\text{CALLS} \leq 2 \cdot \#\text{LEAVES}$$

AND THE LEAVES ARE THE BASE CASE(S): $L(k)$

$$L(k) = \underbrace{L(k-1)}_{1)} + \underbrace{L(k-|N(v)|)}_{2)} \leq \geq 2$$

$$\leq L(k-1) + L(k-2)$$

IT'S FIBONACCI: $L(k) \sim \Phi^k$, $\Phi = 1.6\ldots$

THE TOTAL COST IS THEN $O(\Phi^k \text{POLY}(N))$

BRANCHING + KERNELIZATION

WE CAN PUT THINGS TOGETHER.

VC COSTS:

$$O(n^k)$$

$$\downarrow \text{KERNELIZATION}$$

$$O(\underbrace{(k^2)^k}_{f(k)} + \text{POLY}(n))$$

$$? \quad \downarrow \text{BRANCHING}$$

$$O(\Phi^{k^2} + \text{POLY}(n))$$

$$=$$

$$O(\underbrace{1.6^{k^2}}_{\text{EVEN THOUGH IT CAN BE EXPONENTIAL IN } n, \text{ IN MOST CASES } k \text{ IS SMALL}} + \text{POLY}(n))$$

TODO $\equiv$ QUESTO E' k, n ?

PS: STO ROBO ↑ È' UN CASO DI FPT? PARAMETRO K?

A SNEAK-PEEK TO APPROXIMATING VC

MORE DETAILS IN A FOLLOWING SECTION

CONSIDER A GRAPH $G=(V,E)$.

BY DEF S IS A VERTEX COVER IF

$$\forall uv \in E.\ u \in S \vee v \in S$$

AND WE THEN HAVE THE DECISIONAL PROBLEM VC_k

$$\exists ?\ S \subseteq V.\ |S| \leq k \wedge S \text{ is a V.C.}$$

WE NOW DEFINE VC^*, THE PROBLEM OF FINDING THE SMALLEST VERTEX COVER OF G

$$VC^* = k_{MIN}.\ \forall S \subseteq V.\ |S| < k \Rightarrow S \text{ IS NOT A VERTEX COVER}$$

HOW WE FIND k_{MIN}?

NOT BY DOING BINARY SEARCH: K IS AT THE EXPONENT.

WE SIMPLY START WITH $k=1$ AND SOLVE, AS SEEN BEFORE, $\langle G, k \rangle$.

IT WORKS, BUT IT IS SLOW.

WE CAN APPROXIMATE THE SOLUTION.

WE DO A 2-APPROXIMATION, A CASE OF A TECHNIQUE CALLED R-APPROXIMATION

WE RELAX THE OPTIMALITY: INSTEAD OF LOOKING FOR S^*, WE LOOK FOR A VERTEX COVER $\tilde{S} \subseteq V$ SUCH THAT $|\tilde{S}| \leq \underbrace{2}_{R} \cdot k_{MIN}$

THE PROCEDURE FOR FINDING $\tilde{S}$ IS:

$\tilde{S} = \{\}$

FOR EACH uv IN E:

IF $(u \notin \tilde{S}) \wedge (v \notin \tilde{S})$

$\tilde{S} = \tilde{S} \cup \{u, v\}$

RETURN $\tilde{S}$

AND WE SEE THAT

1) $\tilde{S}$ IS A VC BY DESIGN

2) $|\tilde{S}| \leq 2 \cdot k_{MIN} = 2 \cdot |S^*|$

PROOF:

S^*: BY DEF $\forall uv \in E.\ u \in S^*$ OR $v \in S^*$

CONSIDER $\tilde{S}$: FOR EVERY TWO EDGES THAT "CONTRIBUTES" TO $\tilde{S}$

a) NO ENDPOINT IN COMMON

ALL THE EDGES THAT CONTRIBUTE TO $\tilde{S}$ ARE ABOUT DIFFERENT NODES.

IT CAN'T BE THAT $\underline{uv} \wedge \underline{uy}$ OR ANY OTHER SIMILAR VARIATION.

b) THE #EDGES THAT CONTRIBUTE TO $\tilde{S}$ ARE $\frac{|\tilde{S}|}{2}$ / UNDIRECTED GRAPH

c) EACH OF SUCH EDGE HAS AT LEAST A NODE IN $\tilde{S}$

d) b) + c) $\Rightarrow \frac{|\tilde{S}|}{2} \leq |S^*| \equiv |\tilde{S}| \leq 2|S^*|$ ■

8) RANDOM FPT ALGORITHMS

FPT: FIXED-PARAMETER TRACTABLE

A DESIGN TECHNIQUE FOR SOLVING COMBINATORIALLY HARD (MOSTLY NP-HARD) PROBLEMS.

FOR SOME OF THIS PROBLEMS FPT ACHIEVE BOTH EFFICIENCY AND OPTIMALITY OF THE SOLUTION

EFFICIENCY: $O(f(k) \cdot n^c)$ WITH c A CONSTANT

UNDER FPT FALLS TO APPROACHES:

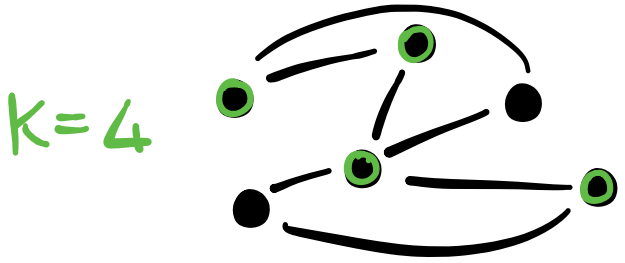
A) COLOR CODING

B) RANDOMIZED SEPARATION } SKIPPED

A) COLOR CODING

CONSIDER AN UNDIRECTED GRAPH $G=(V,E)$

WE DEFINE A K-PATH A PATH OF K NODES WHICH ARE ALL DISTINCT



THE PROBLEM OF FINDING THE K-PATH IS NPC

AS IF $K = N = |V|$ THIS IS HAM.

WE ATTACK THE K-PATH PROBLEM BY COLOR CODING.

CONSIDER A K-COLORING $X: V \to [K]$ THAT ASSOCIATE TO EVERY VERTEX A COLOR.

RANDOMLY!

↳ $\forall v \in V. \forall c \in [K]. P(X(v) = c) = \frac{1}{K}$

OBS. BASICALLY $X \in H$

THEN WE SAY THAT:

A SET $V' \subseteq V$ IS COLORFUL

$\Leftrightarrow$

$\forall u, v \in V'$ WITH $u \neq v$. $X(u) \neq X(v)$

SAID EASY: V' IS COLORFUL IFF EVERY NODE IN V' IS MAPPED TO AN "UNIQUE" COLOR.

WE THEN HAVE THE FOLLOWING PROPOSITIONS:

P.a) A COLORFUL WALK OF K NODES IS A K-PATH

|

- WALK: SEQUENCE OF NODES (AND EDGES) WHERE REPETITION IS ADMITTED.
  PATH: A WALK IN WHICH NEITHER THE NODES NOR EDGES ARE REPEATED
- THE OPPOSITE IS NOT ALWAYS TRUE (OBV, COLORS ARE RANDOMLY ASSIGNED)

P.6) A WALK OF K NODES IS A K-PATH IF AND AND ONLY IF EXISTS A COLORING X S.T THE WALK IS COLORFUL.

CONSIDER $V' \subseteq V$ S.T. $|V'| = k$

$$P(V' \text{ IS COLORFUL}) =$$

$$= \frac{\#\text{COLORINGS } X \text{ THAT MAKES } V' \text{ COLORFUL}}{\#\text{ALL POSSIBLE COLORINGS } X}$$

$$= \frac{k^{N-k} \cdot k!}{k^N}$$

k^{N-k} → IS V/V' ... TODO

k^N → WHY?

$k!$ IS CHOOSE $N, N-1, \ldots$

$$= \frac{k!}{k^k} \cdot \frac{k^{N-k}}{k^N} \equiv \frac{\frac{k^N}{k^k}}{k^N} \equiv \frac{k^N}{k^k} \cdot \frac{1}{k^N} \equiv \frac{1}{k^k}$$

$$\geq e^{-k} \qquad k! \geq \left(\frac{k}{e}\right)^k \text{ KNOWN FACT}$$

$$\Rightarrow \frac{k!}{k^k} \geq \frac{\left(\frac{k}{e}\right)^k}{k^k} \equiv \frac{k!}{k^k} \geq \frac{\frac{k^k}{e^k}}{k^k}$$

$$= \frac{1}{e^k} = e^{-k} \quad \blacksquare$$

QUESTION: GIVEN A RANDOM COLORING X, HOW DO WE FIND COLORFUL PATHS (k-PATHS)?

DYNAMIC PROGRAMMING

a) ANY SUBPATH OF A COLORFUL PATH IS A COLORFUL PATH.

b) A COLORFUL PATH OF LENGTH j ENDS IN A NODE v IF AND ONLY IF EXISTS A NODE $v \in N(v)$ SUCH THAT A COLORFUL PATH OF LENGTH $j-1$ ENDS IN v AND NONE OF THE NODES IN

THAT PATH HAVE A COLOR $X(v)$

c) IN THE COLORFUL PATH OF LENGTH $j-1$ ONLY THE COLORS MATTER, NOT THEIR ORDER OR PATHS.

a) + b) + c) : WE CAN USE DP AND BUILD A TABLE (AKA MATRIX) PATH AS FOLLOWS

$$PATH[S][v] = \begin{cases} 1 \text{ IF } S = \{X(v)\} & \text{TRUE} \\ \text{OR}_{u \in N(v)} PATH[S \setminus X(v)][u] \text{ IF } |S| \geq 2 & \text{b)} \\ \emptyset \text{ OTHERWISE} & \end{cases}$$

IS THERE A COLORFUL PATH THAT ENDS IN v AND THAT USES THE COLORS IN S

THE TABLE PATH HAS $2^k - 1$ ROWS (POWER SET OF k COLORS (S)) AND $n = |V|$ COLUMNS

HENCE THE FINAL RESULTS WILL BE IN THE LAST ROW, WHERE $|S| = k$

EACH ENTRY $PATH[S][v]$ TAKES $O(|N(v)|)$ TIME

TO FILL.

THEN THE TOTAL COST IS:

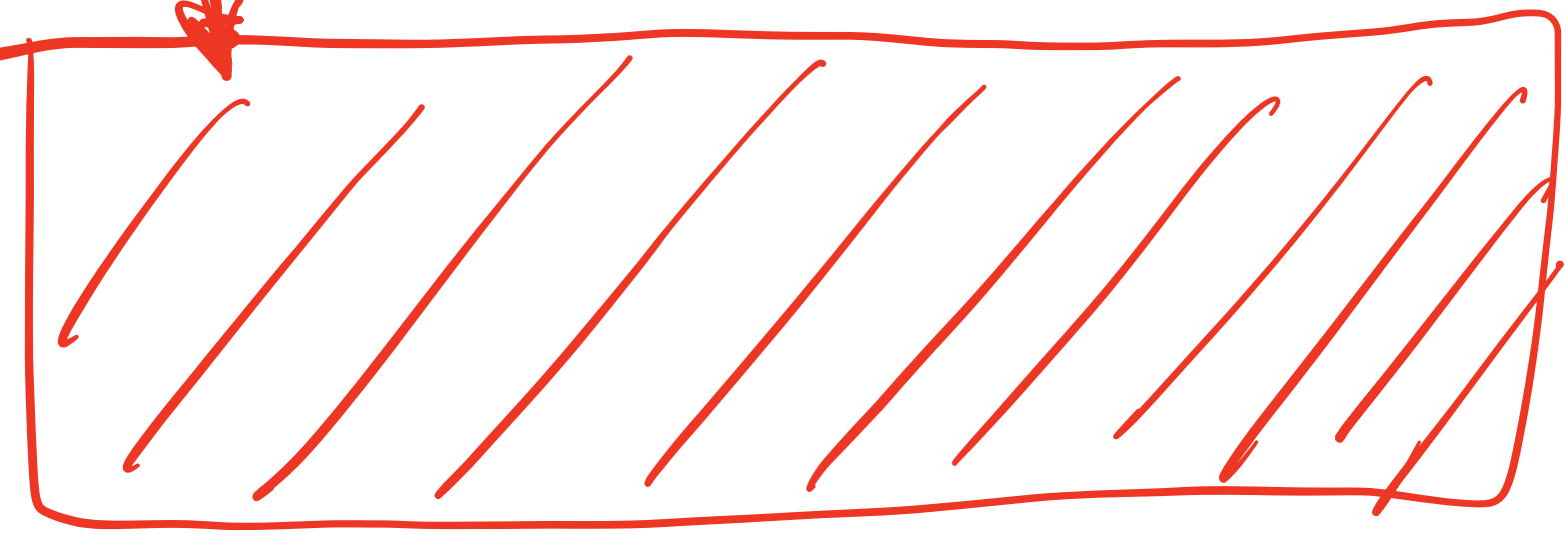
$$\sum_{u \in V} |N(u)| =$$

TODO

$= O(M)$ PER ROW, $M = |E|$

$= O(2^k \cdot M)$

$= O(2^k \cdot N^2)$



FINAL TOUCH

PROB. OF FIND S COLORFUL, COMPUTED ↑

WE REPEAT THE ABOVE FOR e^{-k} TIMES, EACH TIME RANDOMLY CHOOSING A COLORING X.

WE STOP WHEN WE FIND A 1 IN THE LAST ROW.

WE RECALL THE UNION BOUND: CONSTANT PROBABILITY OF SUCCESS, THE RUNNING TIME IS

$$O(e^{-k} 2^k N^2) = O\left(\frac{2^k}{e^k} N^2\right)$$

$O(f(k) \cdot n^c)$, FPT ■

WHY THE COLOR CODING IS POWERFUL?

TAKE A NODE U AND CONSIDER ALL THE K-PATHS ENDING IN U.

POTENTIALLY, WE MAY HAVE UP TO N^K SUCH PATHS, AND A SIGNIFICANT FRACTION OF THEM ARE COLORFUL IN X. — PROPOSITION P.a)

FOR EACH COLORFUL K-PATH, THE SET OF COLOR IS THE SAME, $[K]$

WITH JUST ONE SET WE REPRESENT ALL THE COLORFUL K-PATH.

IN GENERAL: FOR A J-PATH, WHERE $J \in [K]$, WE HAVE $\binom{K}{J}$ SET OF COLORS TO REMEMBER, INSTEAD OF $O(N^J)$ J-PATHS

---

9) R-APPROXIMATION

HARDNESS

- P: PROBLEMS THAT CAN BE SOLVED IN POLYTIME

- NP: PROBLEMS THAT ADMIT A POLY-TIME CERTIFICATE
- NP-HARD: PROBLEMS THAT ARE AT LEAST AS HARD AS THE HARDEST PROBLEM IN NP
- NP-COMPLETE: PROBLEMS BOTH IN NP AND NP-HARD

TO ATTACK SUCH HARD PROBLEMS:

a) EXACT SOLUTIONS: FPT

WE HAVE SEEN K-PATH

b) APPROXIMATE SOLUTIONS (FOR OPTIMIZATION PROBLEMS)

- MIN-COST
- MAX-COST

A TECHNIQUE CALLED R-APPROXIMATION,

THE APPROXIMATED SOLUTION $\tilde{S}$ IS SUCH THAT

- $MIN = \frac{COST(\tilde{S})}{OPT} \leq R$

- $MAX = \frac{OPT}{COST(S)} \leq R$

WITH $R > 1$.

WE HAVE ALREADY APPLIED THIS TECHNIQUE WITH VERTEX COVER.

---

10) NP-COMPLETE PROBLEMS: TSP

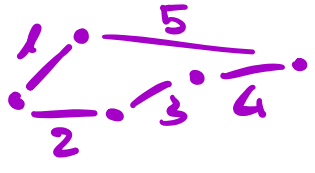
THE TRAVEL SALESMAN PROBLEM

- N CITIES
- D_{IJ}: THE COST/DISTANCE FROM THE CITY i TO THE CITY J
- WE DEFINE A TOUR A PERMUTATION OF ALL THE CITIES

IT'S CLEAR THAT

$$\text{COST OF THE TOUR} = \sum_{i=1}^{N-1} (D_{C_i, C_{i+1}}) + D_{N,1}$$

TOUR: $i, i+1, \ldots, i+N-1, i$



AND WE WANT TO MINIMIZE THE COST.

THEOREM: TSP ∈ NPC

1) THERE IS NO R-APPROX WITH R<1 FOR TSP, UNLESS P=NP.

   ALTERNATIVELY SAID: APPROXIMATING TSP IS AS HARD AS SOLVING TSP

2) IT CAN BE REDUCED IN HAM IN POLY-TIME (SKIPPED)

- METRIC TSP

WE CONSIDER A SUB-VERSION OF TSP AND TO FIND IT WE USE R-APPROX.

IN THIS VERSION OF THE PROBLEM THE GRAPH $G=(V,E)$ IS COMPLETE BY HYP. AND THE EDGES RESPECT THE TRIANGLE INEQUALITY: T.I

$$D_{ij} + D_{jk} \geq D_{ik} \equiv D_{ik} \leq D_{ij} + D_{jk}$$

$\forall u,v \in V. \exists uv \in E$

$G=(V,E,D)$ — SET OF COSTS FOR EVERY EDGE

G IS WEIGHTED

THE ALGORITHM TO FIND THE SOLUTION IS THIS:

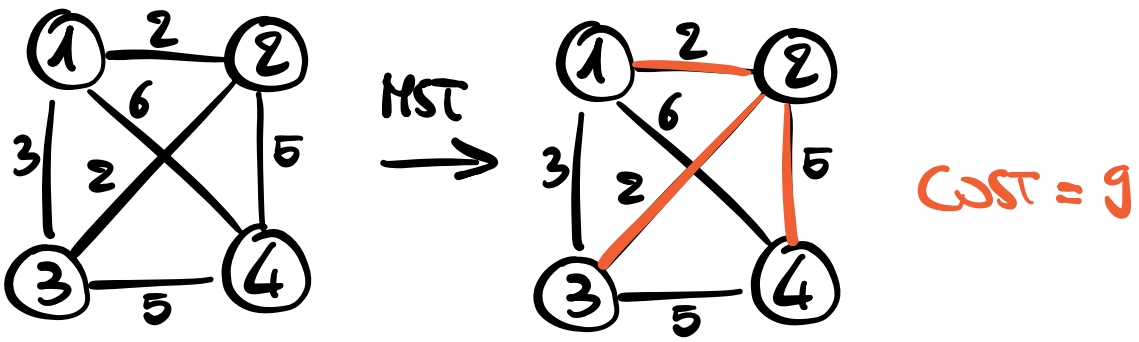
1) CONSIDER $G=(V,E,D)$ COMPLETE, AND THE NODE U FOR WHICH WE WANT THE TSP

2) COMPUTE S = MINIMUM SPANNING TREE OF G (*)
$= MSG(G)$ } WHERE U IS THE ROOT FOR US.

3) COMPUTE THE TOUR T: PERFORM A PREORDER TRAVERSAL OF S: EACH TIME WE SEE A NODE IN S FOR THE FIRST TIME, WE ADD IT TO T (*)

(*) MINIMUM SPANNING TREE

A MINIMUM SPANNING TREE IS A SUBSET OF THE EDGES OF A CONNECTED AND WEIGHTED UNDIRECTED GRAPH, THAT CONNECTS ALL THE VERTICES, WITHOUT ANY CYCLE AND WITH THE MINIMAL POSSIBLE EDGE WEIGHT



* A MST IS A TREE IN ONLY THE SENSE OF LACK OF CYCLES AND BEING CONNECTED.

PICK A ROOT AND SOME ORDER FOR THE SONS, THEN A DFS WILL GIVE A PREORDER TRAVERSAL

EG: WE CHOOSE ① AS A ROOT AND THE ORDER IS THE NUMBERS:

$1 \to 2 \to 3 \to 4 \to 1$ IS THE TSP : T

NOT IN MST: HERE THERE IS THE NECESSITY OF T.I.

AND THIS IS THE POINT!

1) THE COST OF THE OPTIMAL SOLUTION IS GREATER THAN THE COST OF THE MST BY DEF, AS A TOUR WITHOUT AN EDGE IS SURELY A MST

BY DEF. THE OPPOSITE IS NOT AUTOMATIC

2) THE COMPUTED TOUR IS AN APPROXIMATED SOLUTION:

$$\text{COST}(T) \leq 2\,\text{COST}(S)$$

COST OF OUR APPROX. SOLUTION

COST OF THE SPANNING TREE

T. i + EVERY EDGE IS SUMMED TWICE

1) + 2)

$$COST(T) \leq 2 \cdot COST(S) \leq 2 \cdot OPT \quad \blacksquare$$

---

11) NP-COMPLETE PROBLEMS: KNAPSACK

THE KNAPSACK PROBLEM

- N ELEMENTS
- W IS THE KNAPSACK CAPACITY
- w_i IS THE OCCUPANCY OF THE i-TH ELEMENT
- v_i IS THE VALUE OF THE i-TH ELEMENT
- A SOLUTION $S \subseteq [n]$ IS FEASIBLE IF

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

NOTE:

$$\forall i \in N.\ w_i \leq W$$

WE WANT TO MAXIMIZE

$$\sum_{i \in S} v_i$$

THE GREEDY APPROACH

WE DEFINE $\frac{v_i}{w_i}$ AS THE VALUE PER UNIT.

WE DO THE FOLLOWING HYPOTHESIS, WITHOUT LOSS OF GENERALITY:

$$\frac{v_1}{w_1} \geq \dots \geq \frac{v_N}{w_N}$$

TAKE THE i-TH ELEMENT, WITH i AS SMALL AS POSSIBLE, AS SOON AS POSSIBLE

WE CAN FOLLOW THIS GREEDY APPROACH, WHICH GIVES US AN R-APPROXIMATION ($R = 2$):

- $\tilde{S} = \{\}$
- $W' = W$, W' REPRESENTS THE RESIDUAL CAPACITY OF THE KNAPSACK
- $\forall i = 1, 2, \dots, N$:
  - IF $w_i \leq W'$ THEN:
    - $\tilde{S} = \tilde{S} \cup \{i\}$
    - $W' = W' - w_i$
- RETURN $\tilde{S}$

THE PREVIOUS ALGORITHM IS A GOOD APPROXIMATION

BUT IT HAS A GLITCH.

CONSIDER THE FOLLOWING SCENARIO

a) $v_1 = v_2 = \dots = v_{N-1} = 1 \wedge v_N = W - 1$

b) $w_1 = w_2 = \dots = w_{N-1} = 1 \wedge w_N = W$

a) + b) $\Rightarrow$
- $\forall i \in [1, N-1]. \ \frac{v_i}{w_i}$
- $\frac{v_N}{w_N} < 1$

AND WE THEN HAVE

- $\tilde{S} = \{1, \dots, N-1\}$, $\text{COST}(\tilde{S}) = \sum_{i \in S} v_i = N - 1$
- THE OPTIMAL IS $S^* = \{N\}$, $\text{COST}(S^*) = W - 1$

THE APPROXIMATION RATIO IS

$$\frac{OPT}{\text{COST}(\tilde{S})} = \frac{W-1}{N-1} = K$$

K ARBITRARLY LARGE

LET'S FIX THE GLITCH:

WE DEFINE M_G AS THE VALUE RETURNED BY THE GREEDY APPROACH.

LET THEN $V_{MAX} = \max_{i \in [N]} v_i$

AS A SOLUTION TO THE KNAPSACK WE RETURN THE MAXIMUM BETWEEN V_G AND V_{MAX}

THE FIX GIVES US A 2-APPROXIMATION

↳ PROOF:

LET'S DEFINE AN UPPER BOUND TO THE OPT. SOL.

UB OPT

$$UB \geq OPT$$

LET'S THEN DEFINE J AS THE SMALLEST ELEMENT IN [N] THAT DOES NOT FIT IN W' ($J \geq 2$)

IN SYMBOLS:

$$\sum_{i=1}^{J-1} W_i \leq W \text{ BUT } \sum_{i=1}^{J} W_i > W$$

WE THEN USE THE FOLLOWING NOTATION:

- $\overline{W_J} = \sum_{i=1}^{J-1} W_i$
- $\overline{V_J} = \sum_{i=1}^{J-1} V_i$

THEN WE SAY THAT

$$UB = \sum_{i=1}^{J} v_i = \bar{V}_J + v_J > OPT$$

MIND THAT UB IS NOT A SOLUTION, THE OBJECT J CAN'T FIT THE KNAPSACK BY DEF.

SINCE $OPT < \bar{V}_J + v_J$ WE HAVE TWO CASES

1) $\bar{V}_J > v_J \Rightarrow OPT < 2\bar{V}_J \leq 2M_G$

$\bar{V}_J + v_J > OPT + \bar{V}_J > v_J$

$\Rightarrow$

$OPT < 2\bar{V}_J$

AS S IS THE GREEDY SOLUTION IN $\{1, \dots, J-1\} \subseteq S$

2) $\bar{V}_J \leq v_J \Rightarrow OPT < 2v_J \leq 2V_{MAX}$

$v_J \leq V_{MAX}$ BY DEF

SUMMING UP:

$$OPT \leq 2(MAX(M_G, V_{MAX})) = 2 \cdot COST(\tilde{S})$$

$$\Rightarrow$$

$$\frac{OPT}{COST(\tilde{S})} \leq 2 \quad \blacksquare$$

12) FPTAS : FULLY POLYTIME APPROX. SOLUTION

IT'S A ALGORITHM DESIGN TECHNIQUE TO OBTAIN AN R-APPROX FOR ANY $R > 1$

TODO MORE

KNAPSACK FPTAS: DP2 — COME FA? SCALA $1/k$ E UN GREEDY? BOH!

SOLVE AN INSTANCE OF THE KNAPSACK IN

$O(N^2 \cdot V_{MAX})$ TIME ✱ EXACTLY

IDEA: SCALE THE VALUES V_i TO $\tilde{V}_i = \left\lfloor \frac{V_i}{k} \right\rfloor$ FOR SOME SUITABLE FACTOR k TO BE FIXED ASAP.

WE THEN HAVE TWO "ISTANCES" OF THE PROBLEM

a) ORIGINAL: V_i, W_i, W

b) SCALED: $\tilde{V}_i, W_i, W$

S IS FEASIBLE IN a) $\Leftrightarrow$ S IS FEASIBLE IN b

LET'S FIX k SO THAT $\tilde{V}_i$ ARE POLY(N) AS WE WILL PAY $O(N^2 \tilde{V}_{MAX})$

A GOOD CHOICE: ← PERCHÉ?

$$\tilde{V}_i = \left\lfloor \frac{V_i}{k} \right\rfloor \leq \left\lfloor \frac{N}{\varepsilon} \right\rfloor \Rightarrow \frac{V_i}{k} \leq \frac{N}{\varepsilon} \Rightarrow k = \frac{V_{MAX} \cdot \varepsilon}{N}$$

- RUN DP2 ON b) IN TIME $O(N^2 \tilde{V}_{MAX}) = O\left(\frac{N^3}{\varepsilon}\right)$

PERCHÉ?

OBSERVATION: WE FIND THE EXACT OPTIMAL SOLUTION $\tilde{S}$ FOR b)

↓

HOW GOOD $\tilde{S}$ IS FOR a)?

- CLEARLY IT IS FEASIBLE

WHAT ABOUT $\sum_{i \in \tilde{S}} v_i \geq (1-\varepsilon) \sum_{i \in S^*} v_i$

CLEARLY:

* $\sum_{i \in \tilde{S}} \tilde{v}_i \geq \sum_{i \in S^*} \tilde{v}_i$ PERCHÉ?

WE THEN HAVE 3 FACTS:

1) $\forall i . \; v_i \geq \tilde{v}_i$

2) $k \cdot \tilde{v}_i \geq v_i - k$ CHE VUOL DIRE?

3) $\sum_{i \in \tilde{S}} \tilde{v}_i \geq \sum_{i \in S^*} \tilde{v}_i$ PERCHÉ? CHE VUOL DIRE?

THEN: IN THE CONTEXT OF THE VERSION a) OF THE PROBLEM

$$COST(\tilde{S}) =$$

$$= \sum_{i \in \tilde{S}} v_i$$

1) $$\geq \sum_{i \in \tilde{S}} k \tilde{v}_i$$

3) $$\geq \sum_{i \in S^*} k \tilde{v}_i$$

2) $$\geq \sum_{i \in S^*} (v_i - k)$$

$$= \sum_{i \in S^*} v_i - \sum_{i \in S^*} k$$

$|S^*| \leq N$

$$\geq OPT - Nk$$

$Nk = \left(\frac{\varepsilon \cdot v_{MAX}}{N}\right) N = \varepsilon \cdot v_{MAX} \leq \varepsilon \cdot OPT$

$$\geq OPT - \varepsilon\, OPT$$

$$= (1-\varepsilon)\, OPT \quad \blacksquare$$