REOP - Session 3

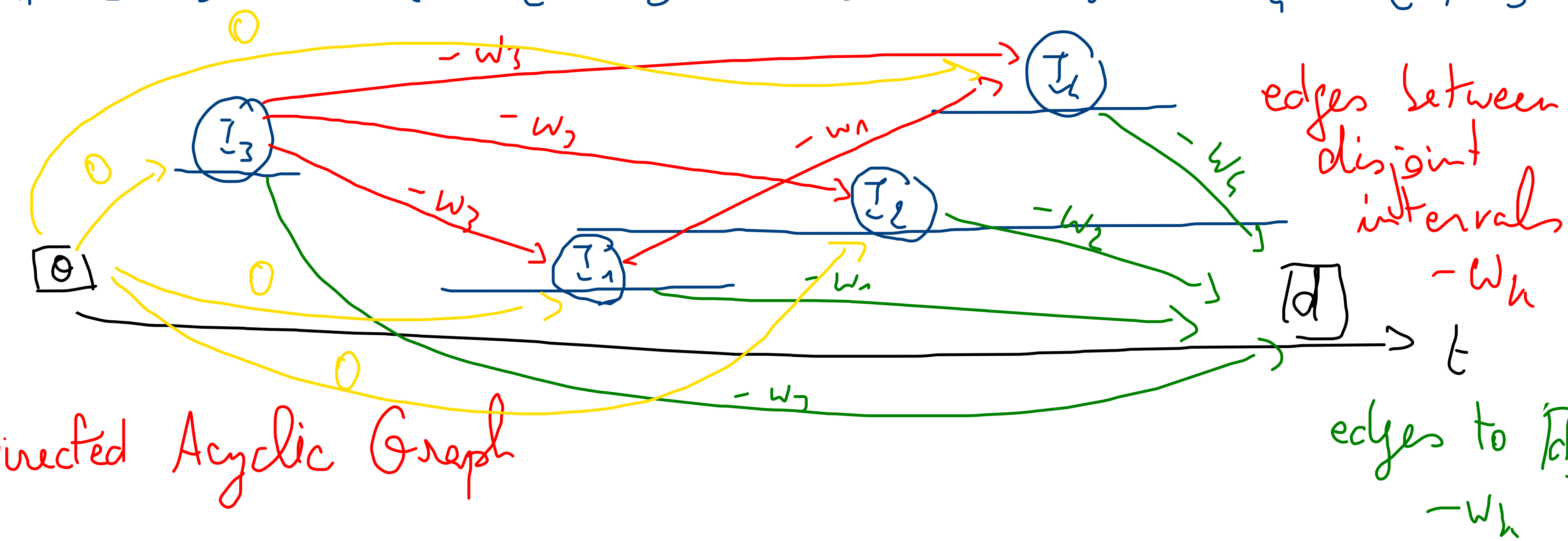
Flows ♡

I/ Homework

A) Ex 5.12 : Flat rental

We have a collection $\mathcal{C}$ of bounded closed intervals $I_k = [a_k, b_k]$ each of them having weight w_k

1) $I_1 = [2,4]$ $I_2 = [3,7]$ $I_3 = [0,1]$ $I_4 = [5,6]$



Directed Acyclic Graph

We look for a shortest path in a graph with negative weights
~~Dijkstra~~ → topological ordering works anyway

A shortest path is a set of disjoint intervals with max weight
(min negative weight)

2) AirBNB host wanting to plan bookings | Disjoint bookings!
Interval = dates of the guest's stay | Maximize your
Weight = price they will pay | revenue! Q1

Q: What if we shift the weights to $\mathbb{R}_+$?
Then we get biased towards paths with
smaller number of edges

3) Ex 5.19 : Longest common subword

w_1 = a b c c b d a c   common subword bcda of length 4
w_2 = b c a d e a

Naïve method: enumerate all subwords of w_1 & w_2 + compare

Complexity: $2^{n_1} \times 2^{n_2} \times \min(n_1, n_2)$ exponential

↳ nb of subwords in w_1 (= subsets of letters) → worst-case comparison

2 nested loops

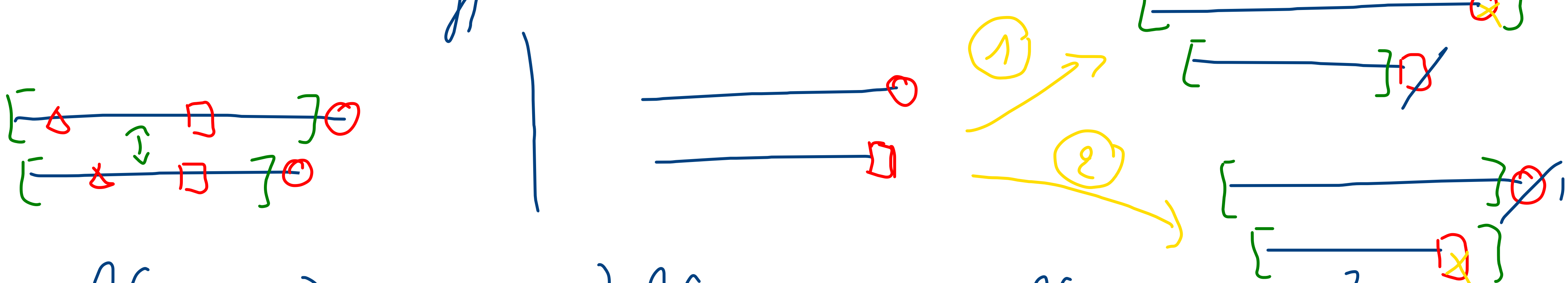
Find a dynamic programming approach: Bellman equation

Generalize the problem:

Compute $\ell(i,j)$ = length of the longest common subword between $w_1[1:i]$ & $w_2[1:j]$

2 cases:

- Last letters are identical: $\ell(n_1, n_2) = \ell(n_1 - 1, n_2 - 1) + 1$
- Last letters are different:



(B) $$\ell(n_1, n_2) = \underline{\max}\{\ell(n_1, n_2 - 1), \ell(n_1 - 1, n_2)\}$$

Initialization: $\ell(0, n_2) = \ell(n_1, 0) = 0$

DP alg : compute $\ell(i,j)$ for $(i,j) \in [\![1, n_1]\!] \times [\![1, n_2]\!]$

each application of the Bellman eq has $O(1)$ cost

total complexity $O(n_1 \times n_2)$

This gives us the length, not the word

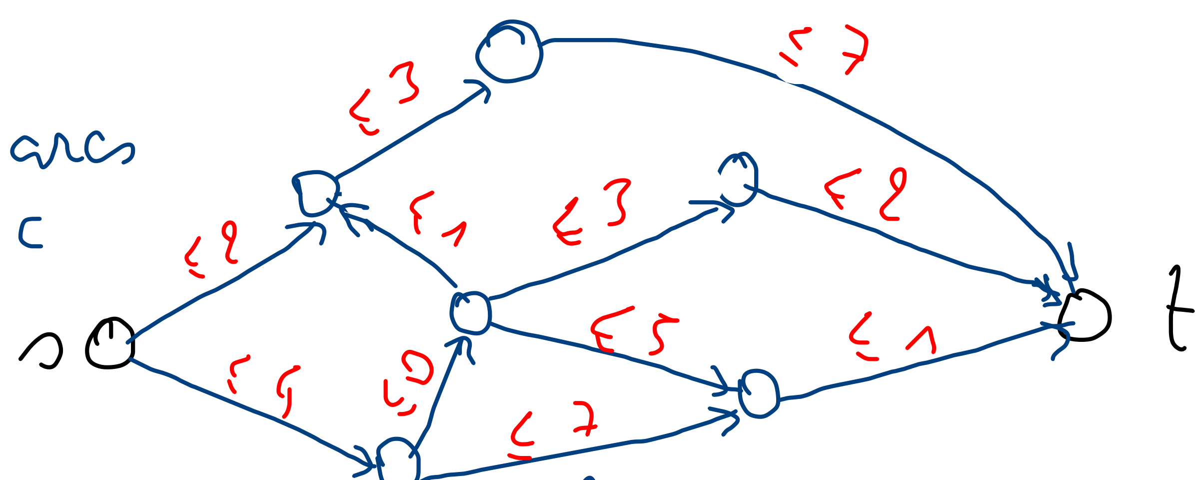
To recover the subword we store the argmax in (B)

II/ Flow vocabulary

A) s-t flow (source - target)

Input : - a digraph $D = (V, A)$ ↳ arcs

- 2 special nodes s & t
- capacities $u(a) \geq 0$ on every arc



An s-t flow is a vector

$f \in \mathbb{R}_+^A$ (s.t. $f(a) \geq 0$)

satisfying 2 constraints : capacity & Kirchhoff

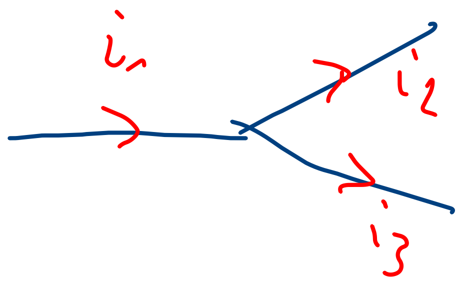
Capacity constraint: $\forall a \in A,\ f(a) \leq u(a)$

Kirchhoff constraint: $\forall v \in V,\ \sum_{a \in \delta^-(v)} f(a) = \sum_{a \in \delta^+(v)} f(a)$

conservation of the flow

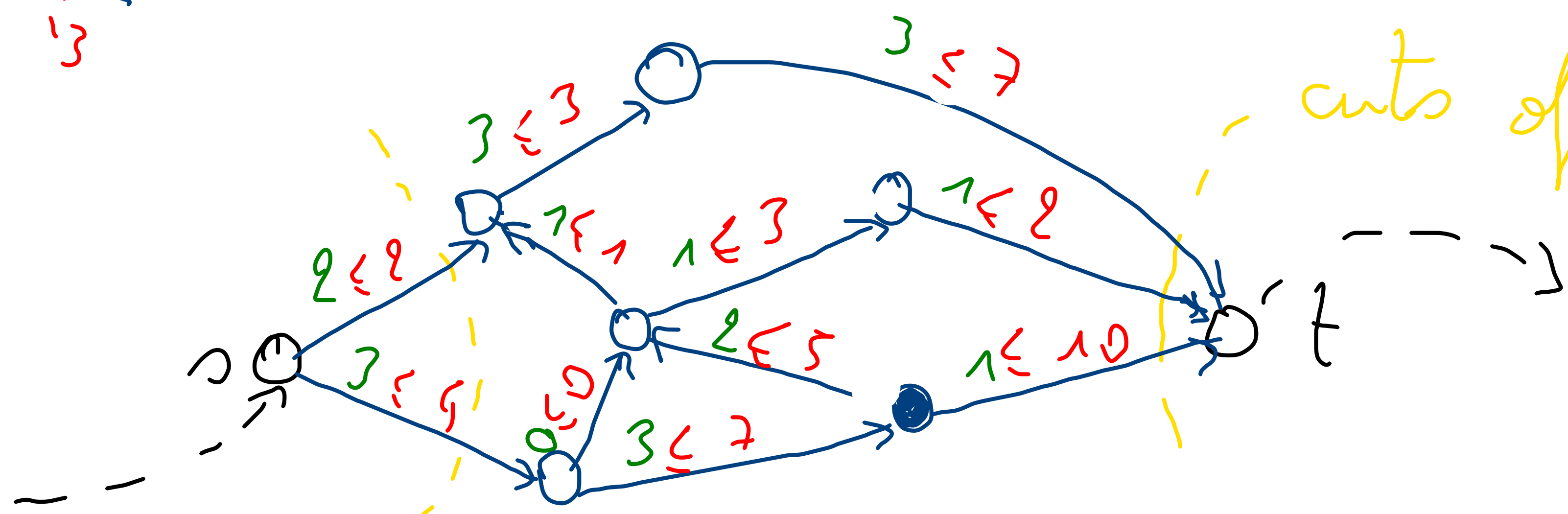
$\delta^-(v)$ ↳ in-neighbors of v

except for s & t



$i_1 = i_2 + i_3$

f



The value of an s-t flow is the total quantity flowing out of the source

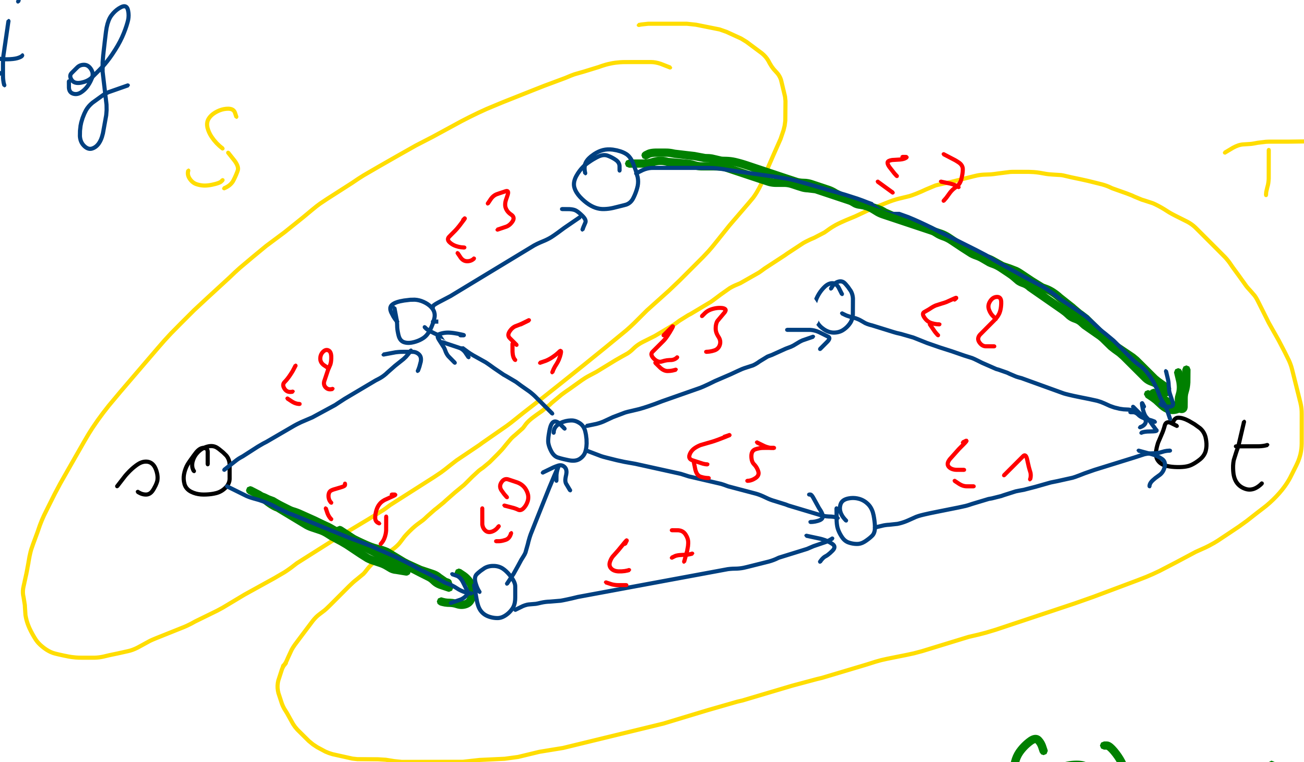
$$val(f) = \sum_{a \in \delta^+(s)} f(a) - \sum_{a \in \delta^-(s)} f(a) = (3+2) - 0$$

B) s-t cuts

An s-t cut is a partition (S, T) of the vertices such that $s \in S$ & $t \in T$

$$\begin{cases} S \cup T = V \\ S \cap T = \emptyset \end{cases}$$

It can also be viewed as a set of edges $B = \delta^+(S)$ splitting the graph between s & t



$u(B) = 4 + 7$

The capacity of the cut is

$$u(S,T) = \sum_{\substack{i \in S, j \in T \\ (i,j) \in A}} u(i,j)$$

$$u(B) = \sum_{a \in B} u(a)$$

2 problems:
- Maximum flow: find an s-t flow of max value
- Minimum cut: find an s-t cut of min capacity

equiv.

Th 6.5: The value of a max flow is equal to the capacity of a min cut

C) Minimum cost flows (other framework, more general)

Input:
- a digraph $D = (V, A)$ (with no "special vertices" s & t)
- lower & upper capacities $0 \leq l(a) \leq u(a)$ on each arc
- cost values $c(a) \geq 0$ on each arc
- inputs / inflows $b(v)$ for each vertex

A b-flow is a vector $f \in \mathbb{R}_+^A$ satisfying

- Capacity constraint: $\forall a \in A,\ \ l(a) \leq f(a) \leq u(a)$
- Kirchhoff constraint: $\forall v \in V,\ b(v) + \sum_{a \in \delta^-(v)} f(a) = \sum_{a \in \delta^+(v)} f(a)$

We must have $\sum_{v \in V} b(v) = 0$

algebraic
≥ 0 : flows in
≤ 0 : flows out

Problem: find a b-flow with minimum cost

$$c(f) = \sum_{a \in A} c(a) f(a)$$

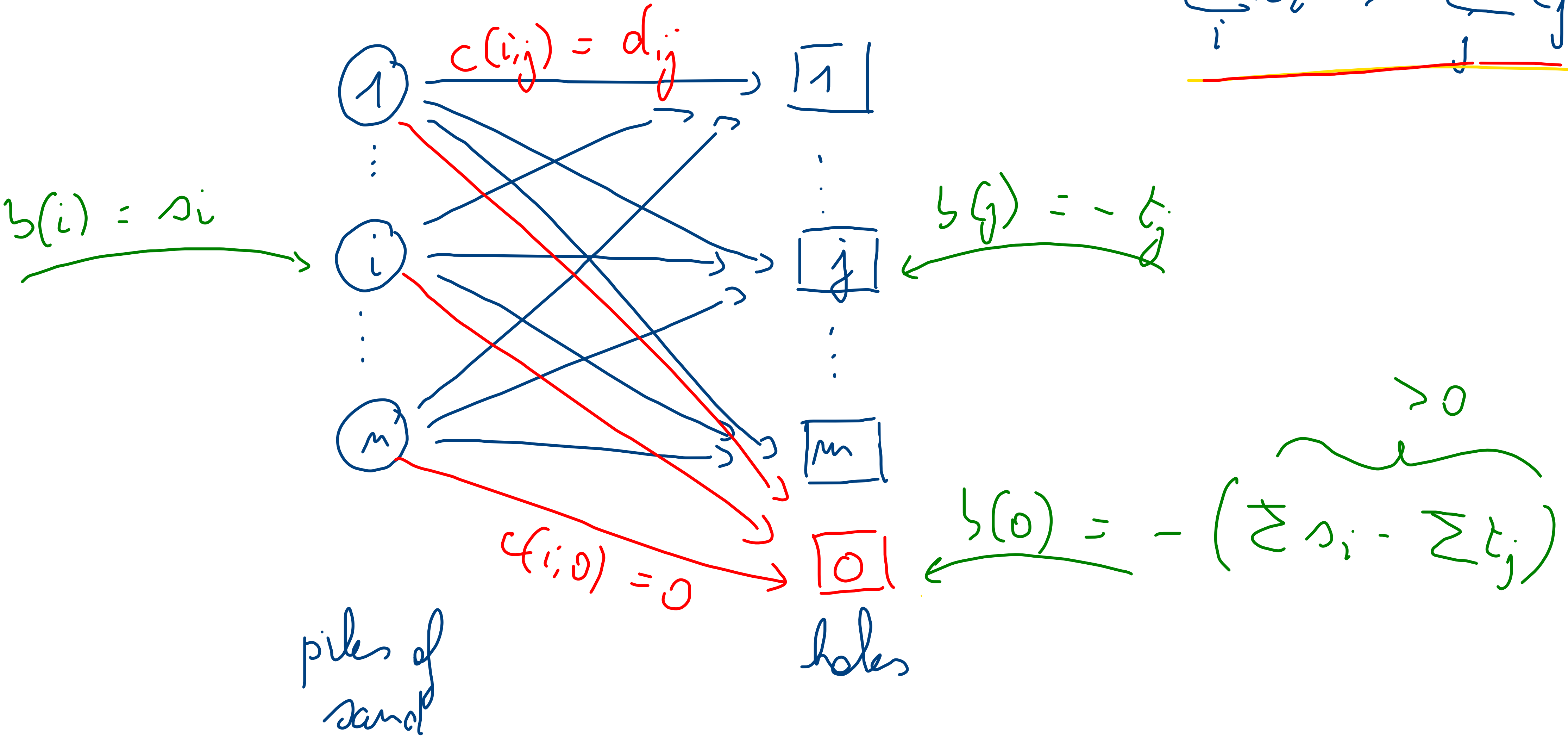
unit cost

D) Exercises 6.9 & 6.13

Ex 6.9 (Monge's transportation pb)

What if

$$\sum_i s_i > \sum_j t_j$$



Flow variable x_{ij} = quantity of sand transported from i to j
Solve a minimum cost flow problem (no capacity constraints)

IV/ Linear Programming for flows

General technique for flow problems
Useful if there additional "nonstandard constraints"

Formulation for max s-t flow:

$$(LP)\quad \max_f \sum_{a\in\delta^+(s)} f_a - \sum_{a\in\delta^-(s)} f_a = val(f) \qquad \text{linear}$$

$$\text{s.t.} \quad \left| \begin{array}{lll} 0 \leq f_a \leq u(a) & \forall a \in A & \text{linear} \\ \displaystyle\sum_{a\in\delta^-(v)} f_a = \sum_{a\in\delta^+(v)} f_a & \forall v \in V \setminus \{s,t\} & \text{linear} \end{array} \right.$$

Ex: do the same for min cost b-flow

Prop 6.11: Integer capacities $\Longrightarrow$ integer flow values at the optimum
Prop 6.12: The dual of (LP) is the minimum cut problem

III/ Algorithms for the max s-t flow problem

Rq: Of course there are similar algorithms for the min cost b-flow (see notes)

A) Optimality condition

Prop 6.3: If f is an s-t flow & (S,T) is an s-t cut, then

$$\mathrm{val}(f) \leq u(S,T)$$

Proof:

$$\mathrm{val}(f) = \sum_{a\in\delta^+(s)} f(a) - \sum_{a\in\delta^-(s)} f(a) + \sum_{\substack{v\in S\\ v\neq s}} \overbrace{\left(\sum_{a\in\delta^+(v)} f(a) - \sum_{a\in\delta^-(v)} f(a)\right)}^{=0 \text{ by conservation}}$$

$$= \sum_{a\in\delta^+(S)} f(a) - \sum_{a\in\delta^-(S)} f(a)$$

edges within S cancel out

capacity

$$\leq \sum_{a\in\delta^+(S)} u(a) - \cancel{\sum_{a\in\delta^-(S)} 0}$$

$$= \underbrace{\qquad}_{u(S,T)}$$

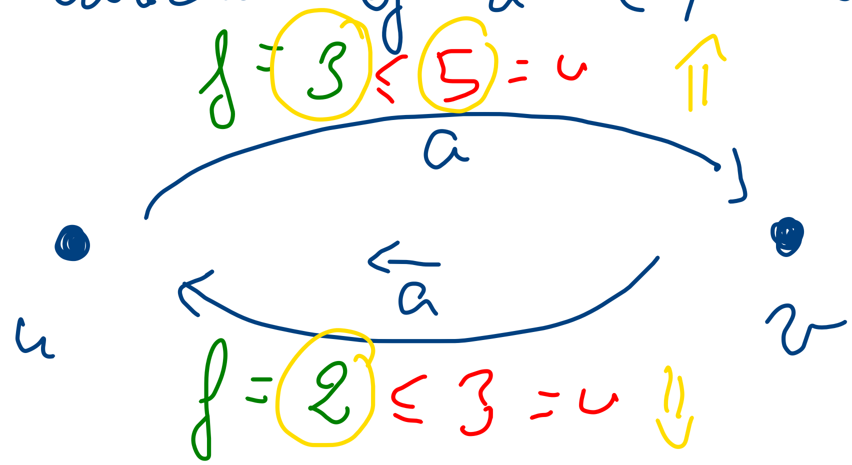
For every arc $a = (u, v)$ we define $\overleftarrow{a} = (v, u)$ and residual capacities

along $\overleftarrow{a}$

$$u_r(a) = (u(a) - f(a)) + (f(\overleftarrow{a}))$$

along a

They tell us how much we can increase the flow in the direction of a ($\neq$ along the arc a)



2 ways to ↑ the flow

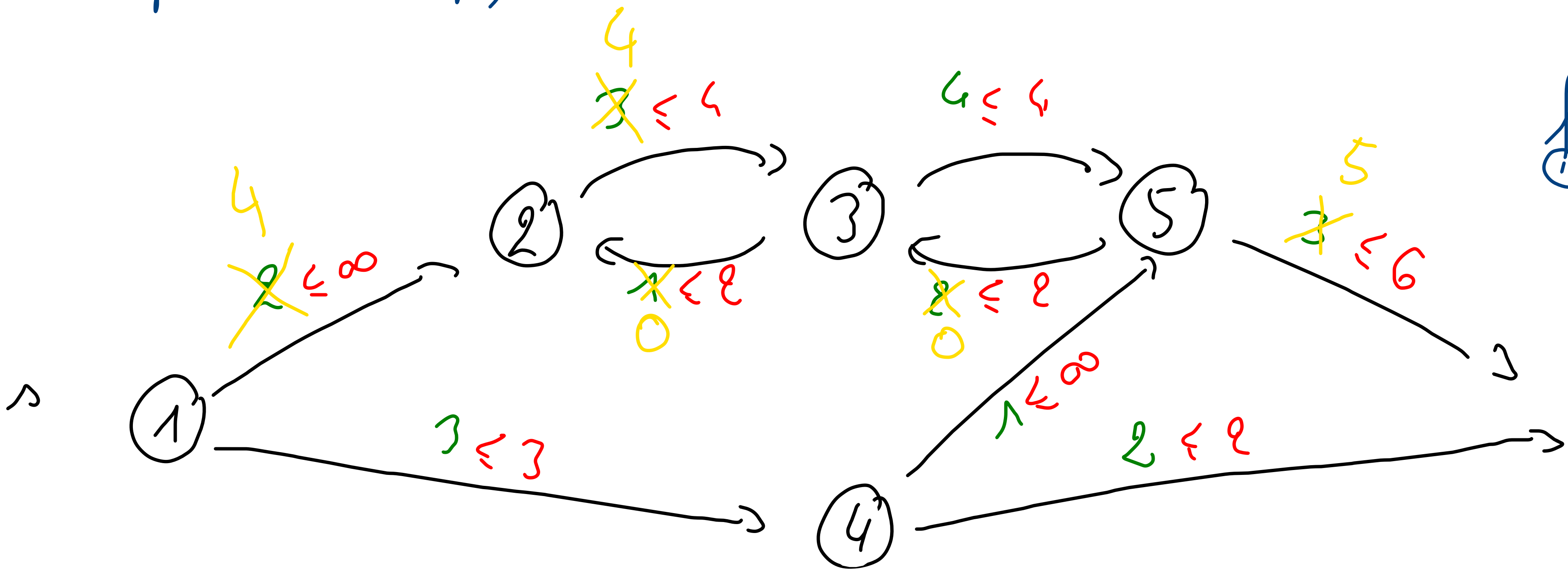
- more flow on a $+(5-3)$
- less flow on $\overleftarrow{a}$ -2

The residual graph $D_r = (V, A_r)$ is the capacited graph with

$$A_r = \{ a \in A \cup \overleftarrow{A} : u_r(a) > 0 \}$$

An augmenting path P is an s-t path in the residual graph

Example (Wooclap)

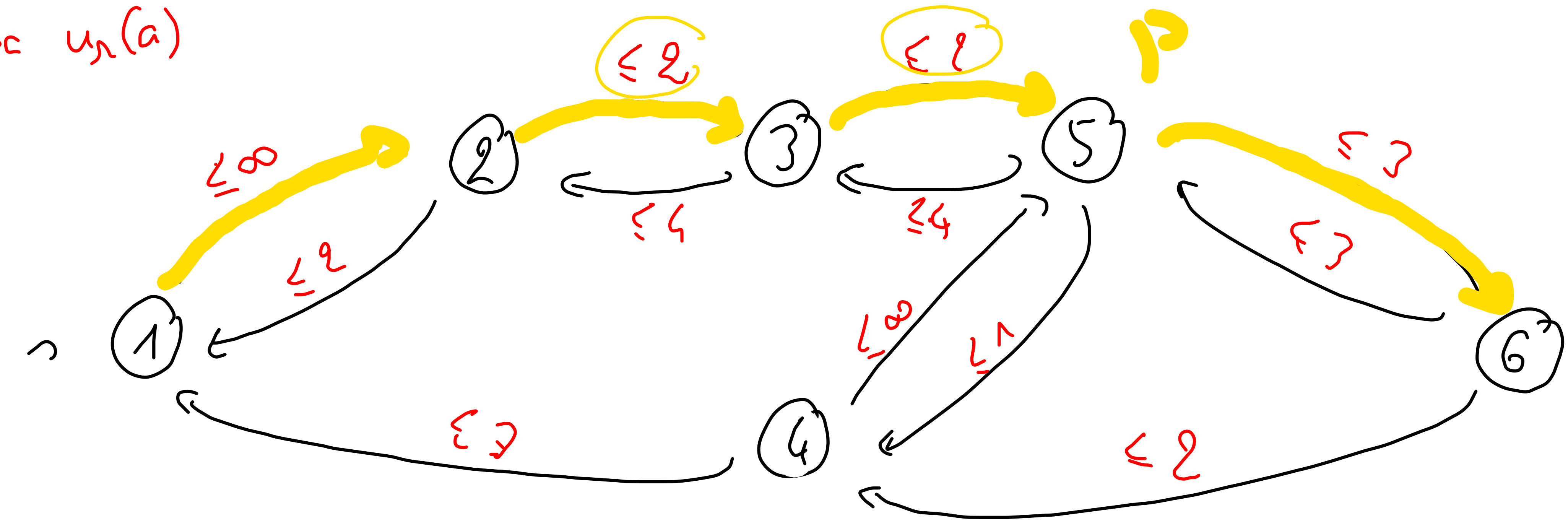


flow graph f

val(f) = ~~5~~ 7

residual graph

capa $u_f(a)$



augmenting path P

capa 2

If we rebuild the residual graph, s & t are now in separate connected components, so there is no augmenting path in D_r

Thm 6.4 : An s-t flow is maximal iff there is no augmenting path in the residual graph

B) Ford-Fulkerson algorithm

1. Start with $f(a) = 0 \quad \forall a \in A$
2. While there is an augmenting path:
   - Select an augmenting path P with min number of edges → Breadth-First Search
   - Increase the flow along P by $\min_{a \in P} u_r(a)$
3. Return f

Questions: - How do we find an augmenting path?
⇒ Path from s to t in D_r
- Which path do we select? Important for complexity

Edmonds-Karp algorithm → polynomial runtime

$O(|A|^2 \times |V|)$ (can be better)

V/ Homework

Ex 6.13, 6.10 if you want: Ex 6.16, 6.18