# Time versus Space

# Time versus Space

- $TIME(t(n)) \subseteq SPACE(t(n))$
- is the inclusion proper?
- intuitively yes
- proof: Hopcroft, Paul and Valiant 1975

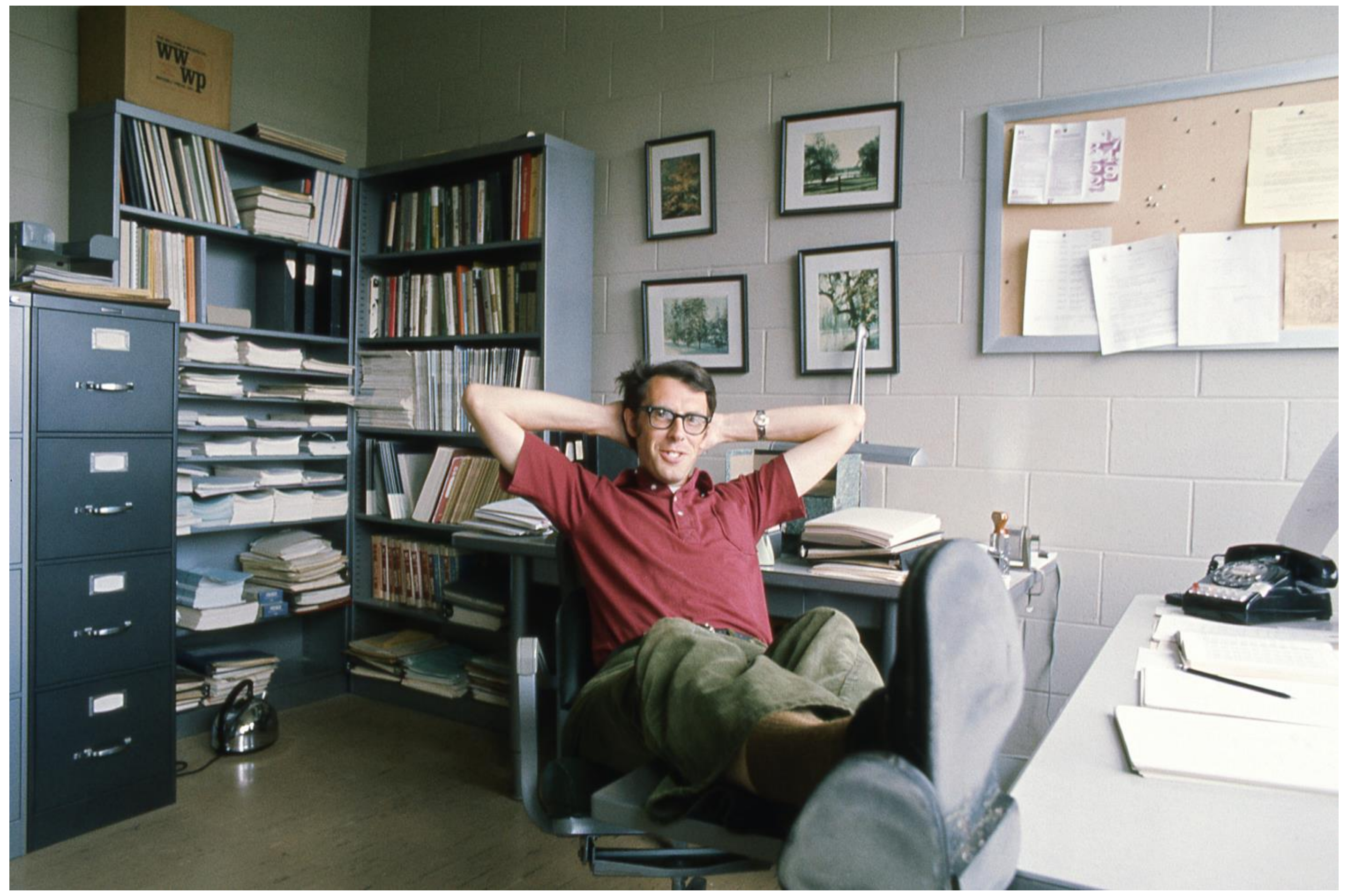


foto:1986

# Time versus Space

- $TIME(t(n)) \subseteq SPACE(t(n))$
- is the inclusion proper?
- intuitively yes
- proof: Hopcroft, Paul and Valiant 1975

**simulation:**

**Lemma 1.**

$$TIME(t(n)) \subseteq SPACE(t(n)/\log(t(n)))$$

**separating complexity classes**

**Lemma 2.** *Let $t(n)$ be space constructible*

$$TIME(t(n)) \subsetneq SPACE(t(n))$$

# Time versus Space

- $TIME(t(n)) \subseteq SPACE(t(n))$
- is the inclusion proper?
- intuitively yes
- proof: Hopcroft, Paul and Valiant 1975

**simulation:**

**Lemma 1.**

$$TIME(t(n)) \subseteq SPACE(t(n)/\log(t(n)))$$

**separating complexity classes**

**Lemma 2.** *Let $t(n)$ be space constructible*

$$TIME(t(n)) \subsetneq SPACE(t(n))$$

$$\begin{aligned} TIME(t(n)) &\subseteq SPACE(t(n)/log(t(n))) \quad \text{(lemma 1 )} \\ &\subsetneq SPACE(t(n)) \quad \text{(space hierarchy theorem)} \end{aligned}$$

# Time versus Space

- $TIME(t(n)) \subseteq SPACE(t(n))$
- is the inclusion proper?
- intuitively yes
- proof: Hopcroft, Paul and Valiant 1975

initial idea:

- all simulations we know are step by step (in order)
- what if we try to simulate out of order?

**simulation:**

**Lemma 1.**

$$TIME(t(n)) \subseteq SPACE(t(n)/\log(t(n)))$$

**separating complexity classes**

**Lemma 2.** *Let $t(n)$ be space constructible*

$$TIME(t(n)) \subsetneq SPACE(t(n))$$

$$\begin{aligned} TIME(t(n)) &\subseteq SPACE(t(n)/log(t(n))) \quad \text{(lemma 1 )} \\ &\subsetneq SPACE(t(n)) \quad \text{(space hierarchy theorem)} \end{aligned}$$

# 1 Pebble game

- played on finite directed acyclic graphs (dag's) $G = (V, E)$
- by placing pebbles on nodes or removing pebbles from nodes
- legal moves (illustrated in figure 1 )
  1. placing pebble on a source of the graph (reading input)
  2. if all direct ancestors of a node $u$ have a pebble, place a pebble on $u$
  3. remove a pebble (free space from intermediate result)

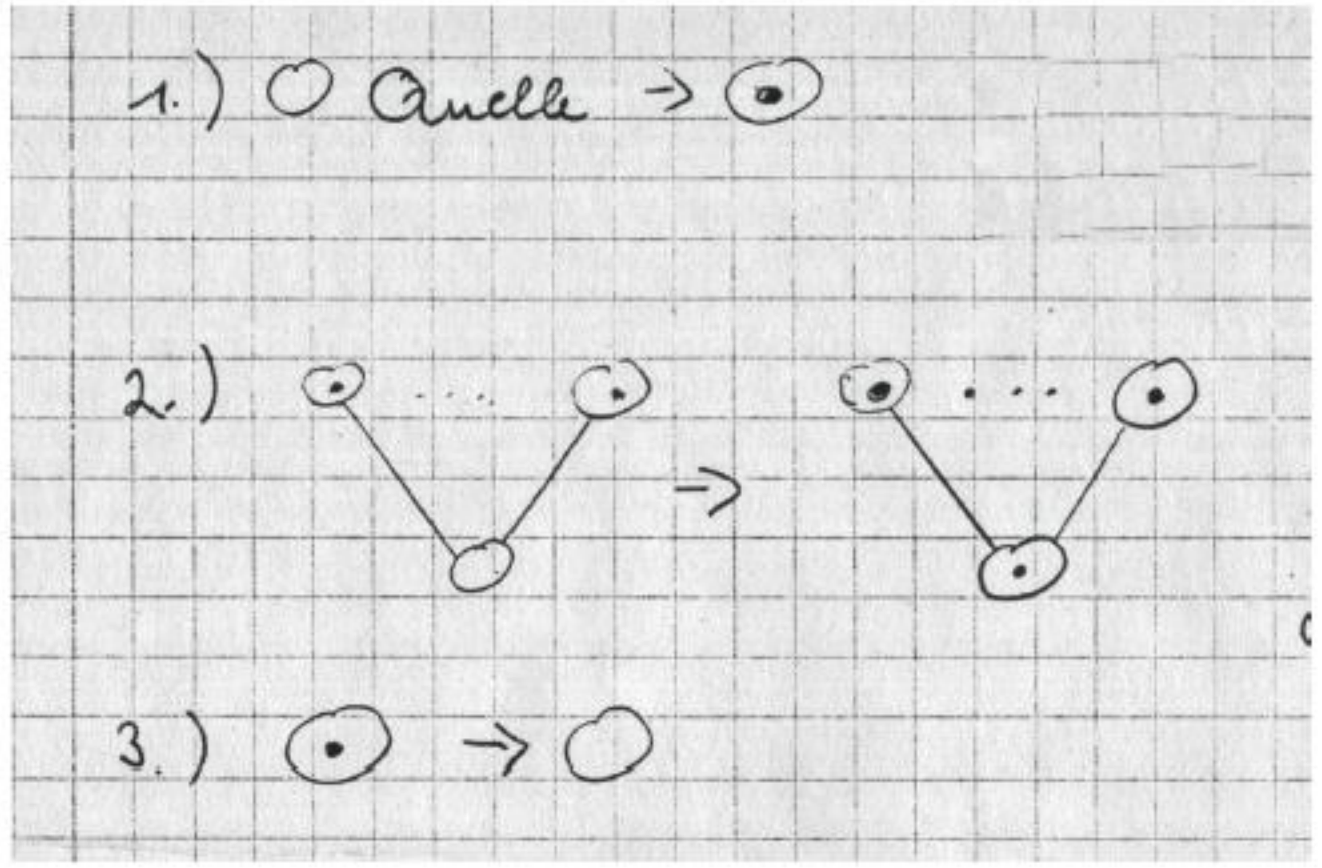



Figure 1: Legal moves: 1) place a pebble on a source 2) place a pebble on a node if all direct predecessors have pebbles 3) remove pebbly any time anywhere

# 1 Pebble game

- played on finite directed acyclic graphs (dag's) $G = (V, E)$
- by placing pebbles on nodes or removing pebbles from nodes
- legal moves (illustrated in figure 1 )
  1. placing pebble on a source of the graph (reading input)
  2. if all direct ancestors of a node $u$ have a pebble, place a pebble on $u$
  3. remove a pebble (free space from intermediate result)

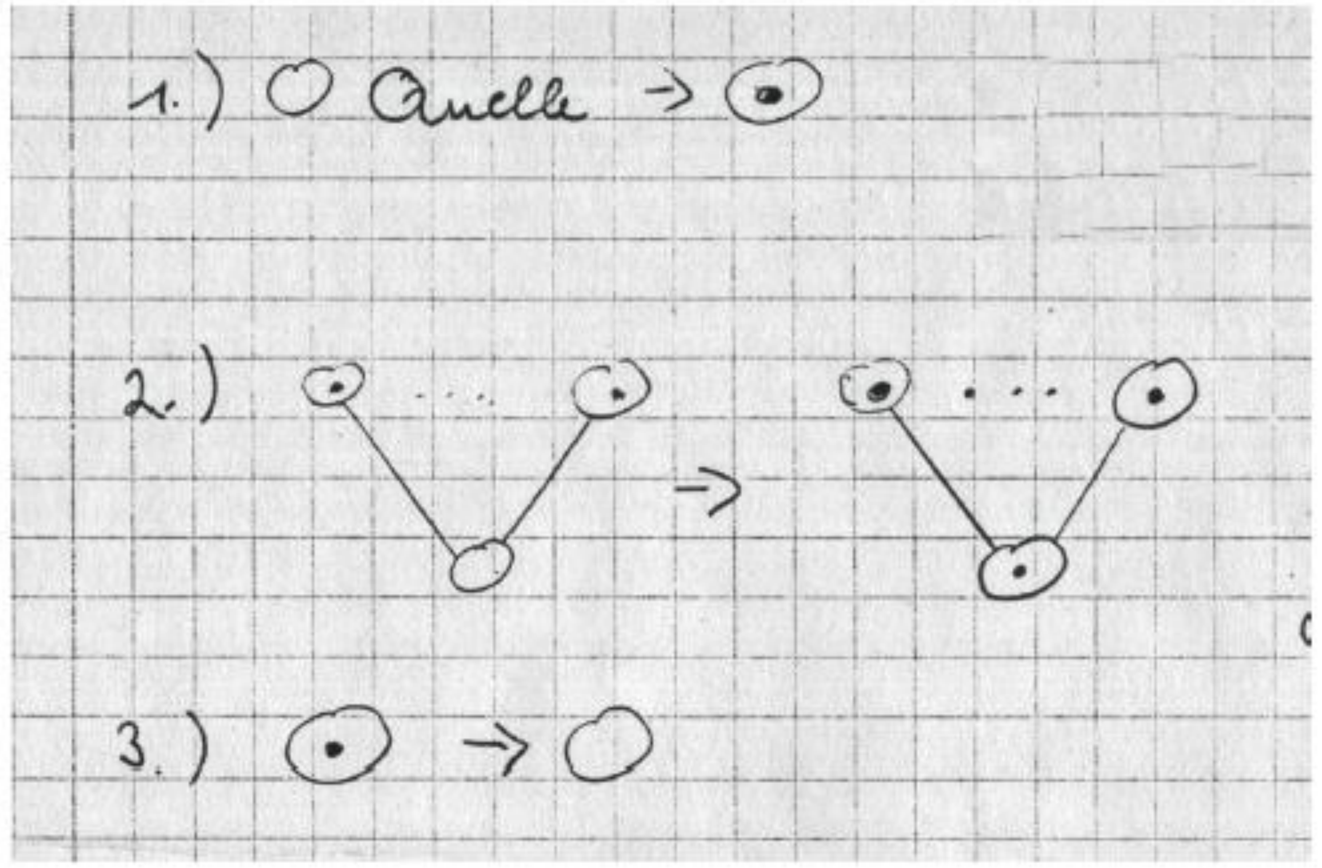



Figure 1: Legal moves: 1) place a pebble on a source 2) place a pebble on a node if all direct predecessors have pebbles 3) remove pebbly any time anywhere

- goal: to *pebble* the graph, i.e.
  1. place a pebble on each node of the graph at some time
  2. using as few pebbles as possible.

# 1 Pebble game

- played on finite directed acyclic graphs (dag's) $G = (V, E)$
- by placing pebbles on nodes or removing pebbles from nodes
- legal moves (illustrated in figure 1 )
  1. placing pebble on a source of the graph (reading input)
  2. if all direct ancestors of a node $u$ have a pebble, place a pebble on $u$
  3. remove a pebble (free space from intermediate result)

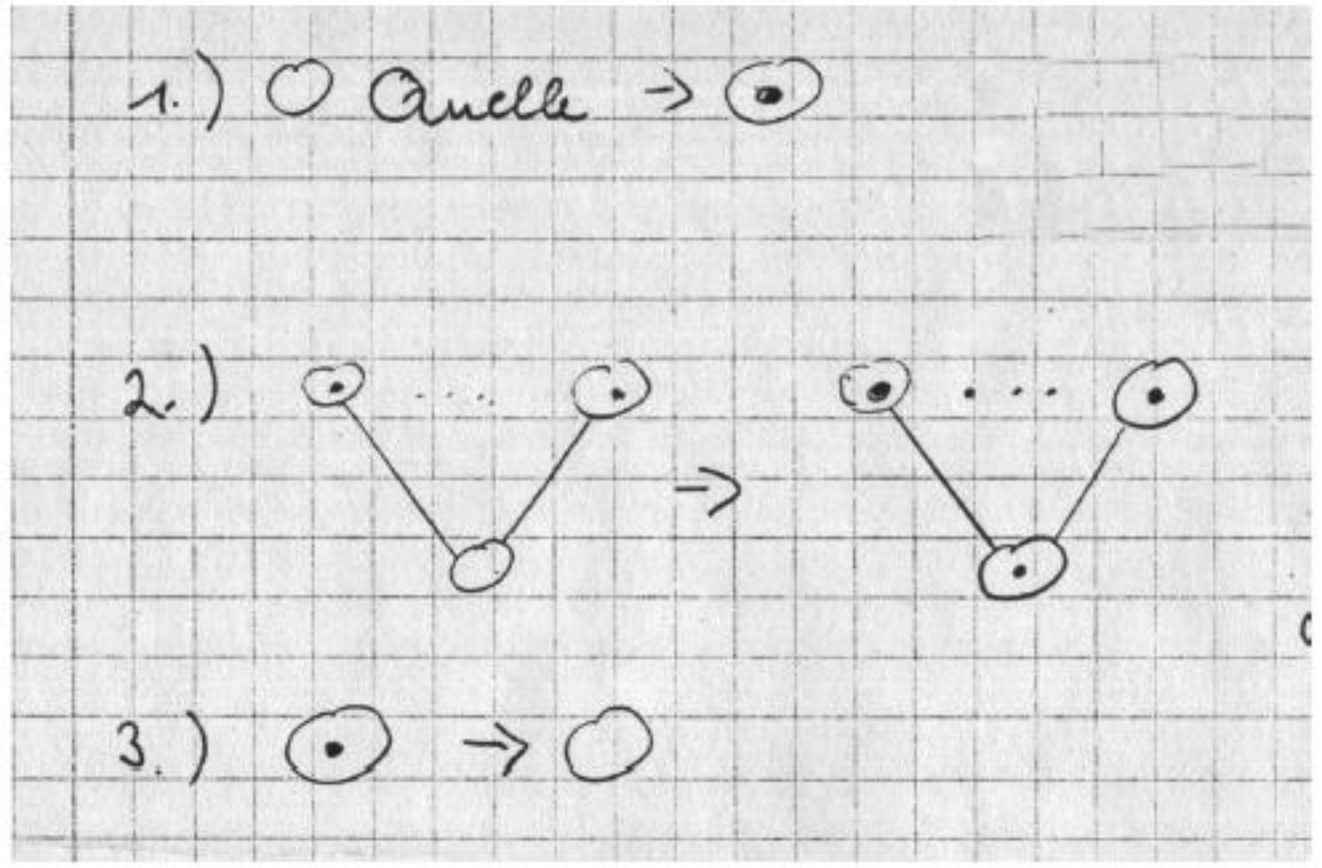



Figure 1: Legal moves: 1) place a pebble on a source 2) place a pebble on a node if all direct predecessors have pebbles 3) remove pebbly any time anywhere

- goal: to *pebble* the graph, i.e.
  1. place a pebble on each node of the graph at some time
  2. using as few pebbles as possible.

I2OS:

**Sethi-Ullman register allocation:** uses this game on derivation trees of expressions

# 1 Pebble game

- played on finite directed acyclic graphs (dag's) $G = (V, E)$
- by placing pebbles on nodes or removing pebbles from nodes
- legal moves (illustrated in figure 1 )
  1. placing pebble on a source of the graph (reading input)
  2. if all direct ancestors of a node $u$ have a pebble, place a pebble on $u$
  3. remove a pebble (free space from intermediate result)

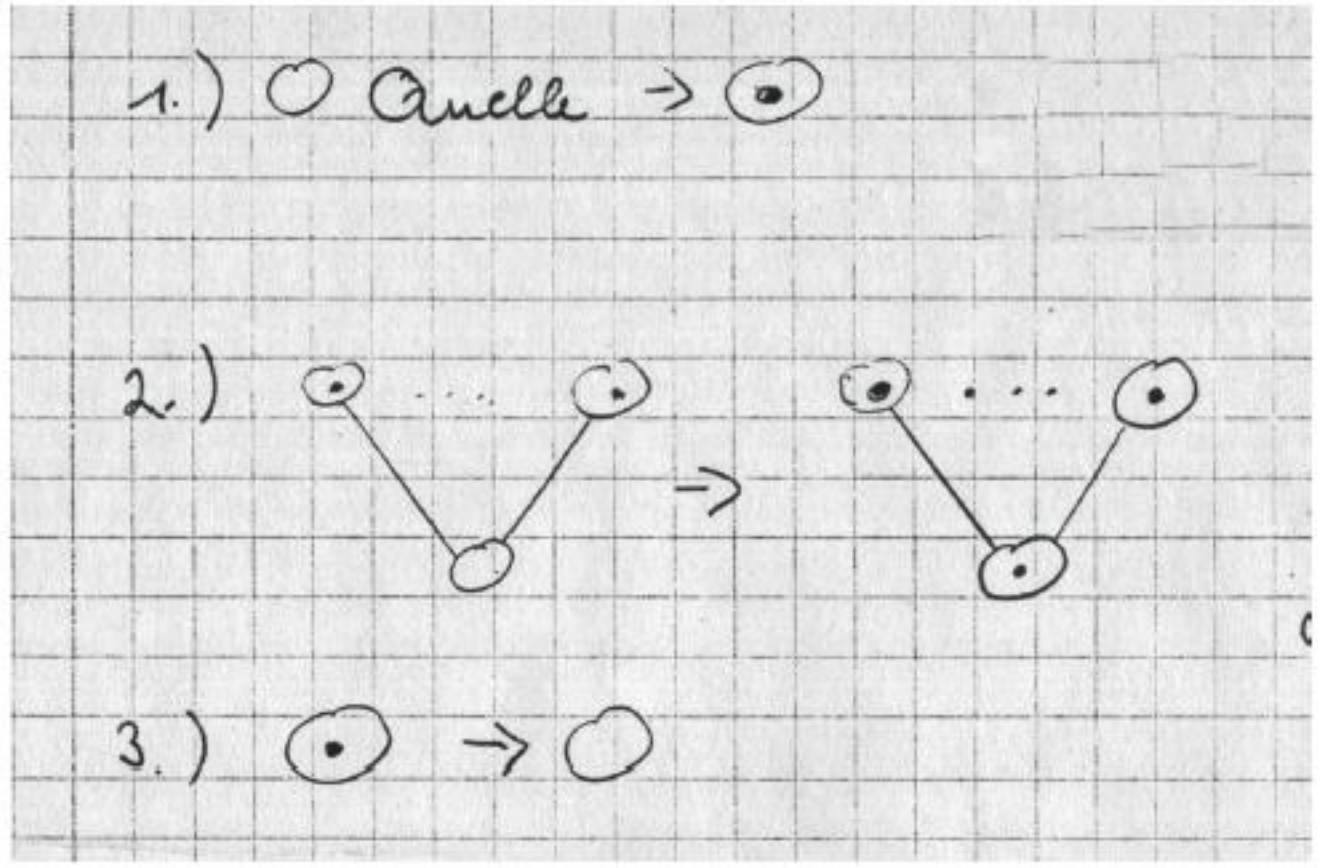


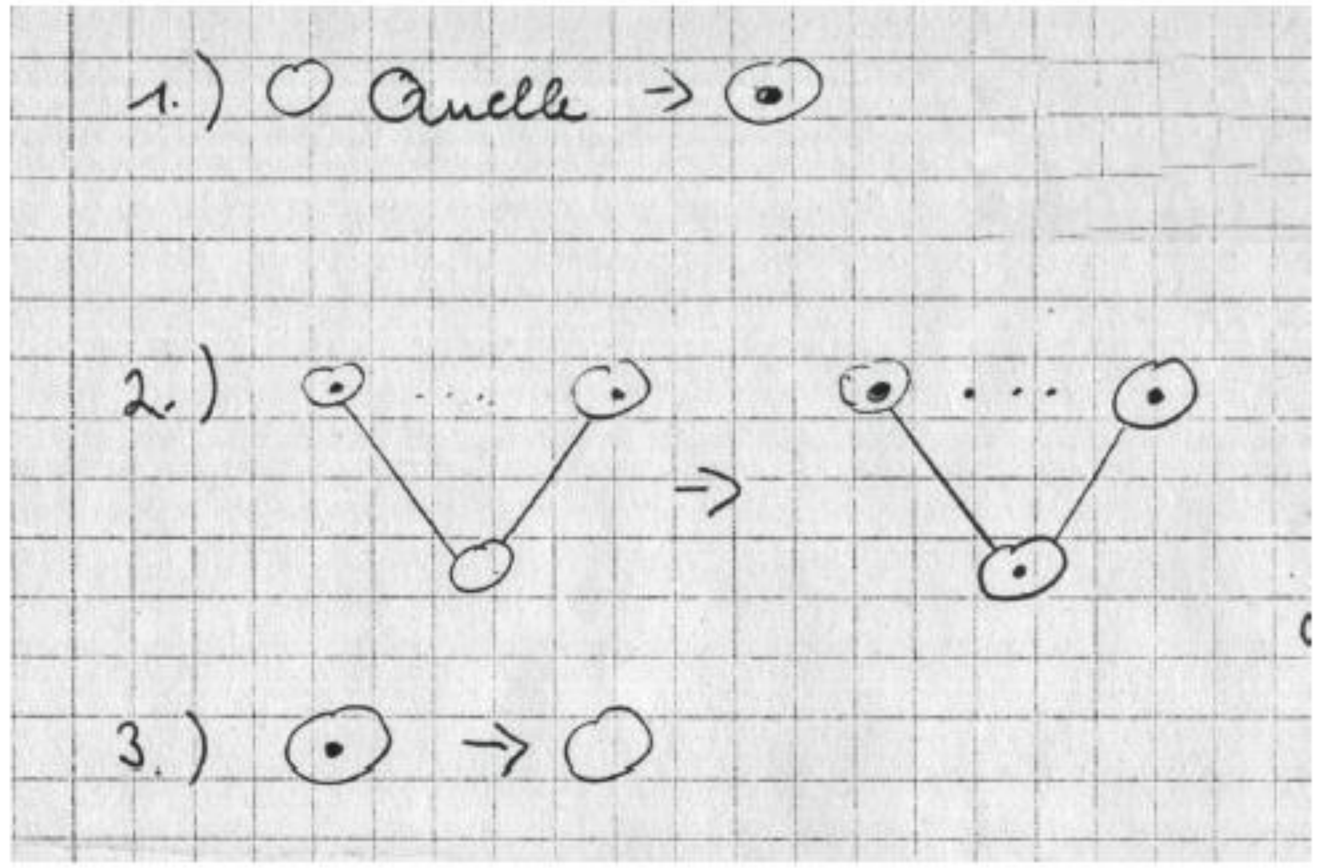

Figure 1: Legal moves: 1) place a pebble on a source 2) place a pebble on a node if all direct predecessors have pebbles 3) remove pebbly any time anywhere

- goal: to *pebble* the graph, i.e.
  1. place a pebble on each node of the graph at some time
  2. using as few pebbles as possible.

I2OS:

**Sethi-Ullman register allocation:** uses this game on derivation trees of expressions

- played on finite directed acyclic graphs (dag's) $G = (V, E)$
- by placing pebbles on nodes or removing pebbles from nodes
- legal moves (illustrated in figure 1 )
  1. placing pebble on a source of the graph (reading input)
  2. if all direct ancestors of a node $u$ have a pebble, place a pebble on $u$
  3. remove a pebble (free space from intermediate result)



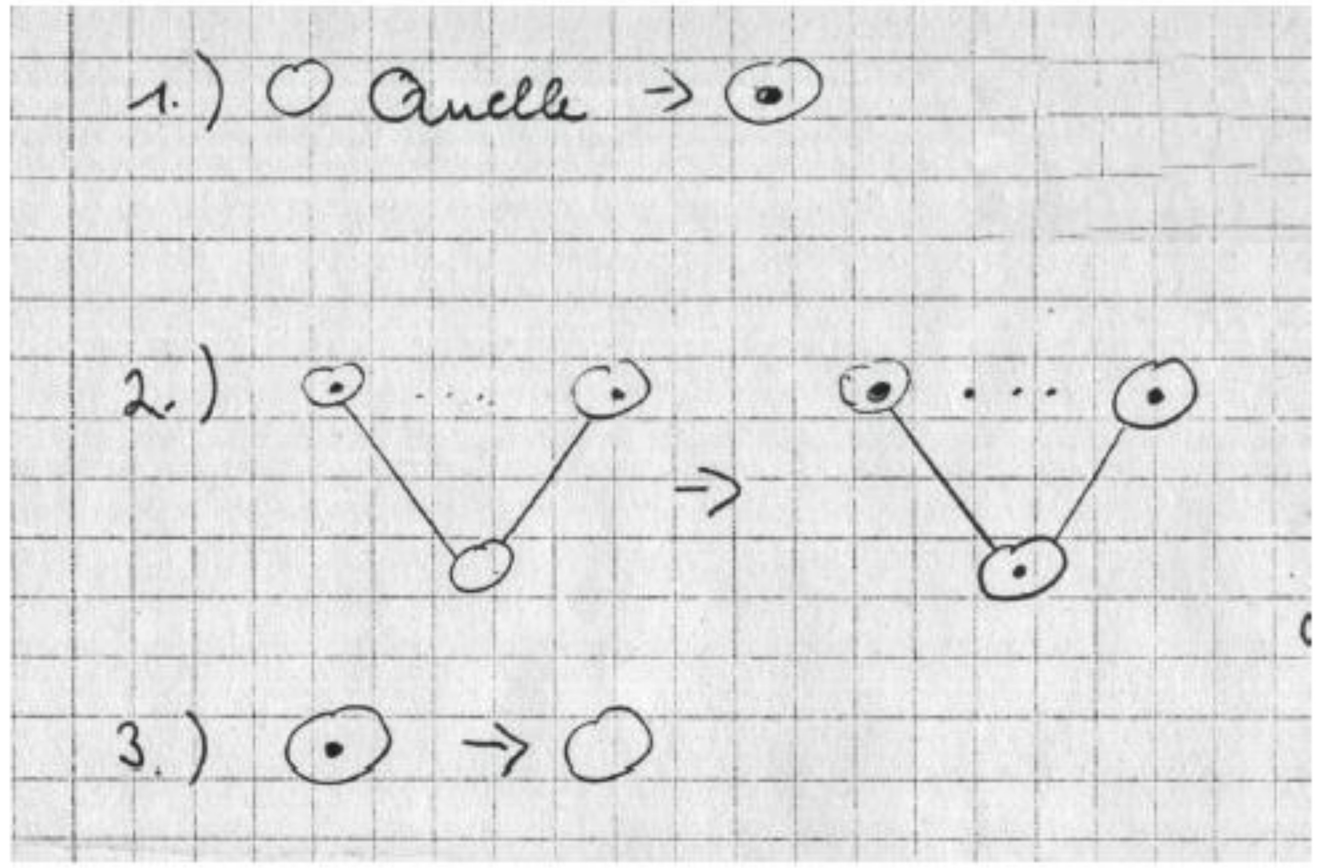

- goal: to *pebble* the graph, i.e.
  1. place a pebble on each node of the graph at some time
  2. using as few pebbles as possible.

# 2 Pebble lemma

**def: indegree of a graph:** maximal number of direct ancestors of any node in the graph

**Lemma 3.** *Let $P_k(n)$ be the smallest number of pebbles sufficient to pebble any dag with $n$ edges and indegree $k$. Then*

$$\begin{aligned} P_k(n) &= O(n/\log n) \\ &\leq C_k \cdot n/\log n \quad \textit{faa } n \end{aligned}$$




**def:** **indegree of a graph:** maximal number of direct ancestors of any node in the graph

**Lemma 3.** *Let $P_k(n)$ be the smallest number of pebbles sufficient to pebble any dag with n edges and indegree k. Then*

$$\begin{aligned} P_k(n) &= O(n/\log n) \\ &\leq C_k \cdot n/\log n \quad \textit{faa } n \end{aligned}$$

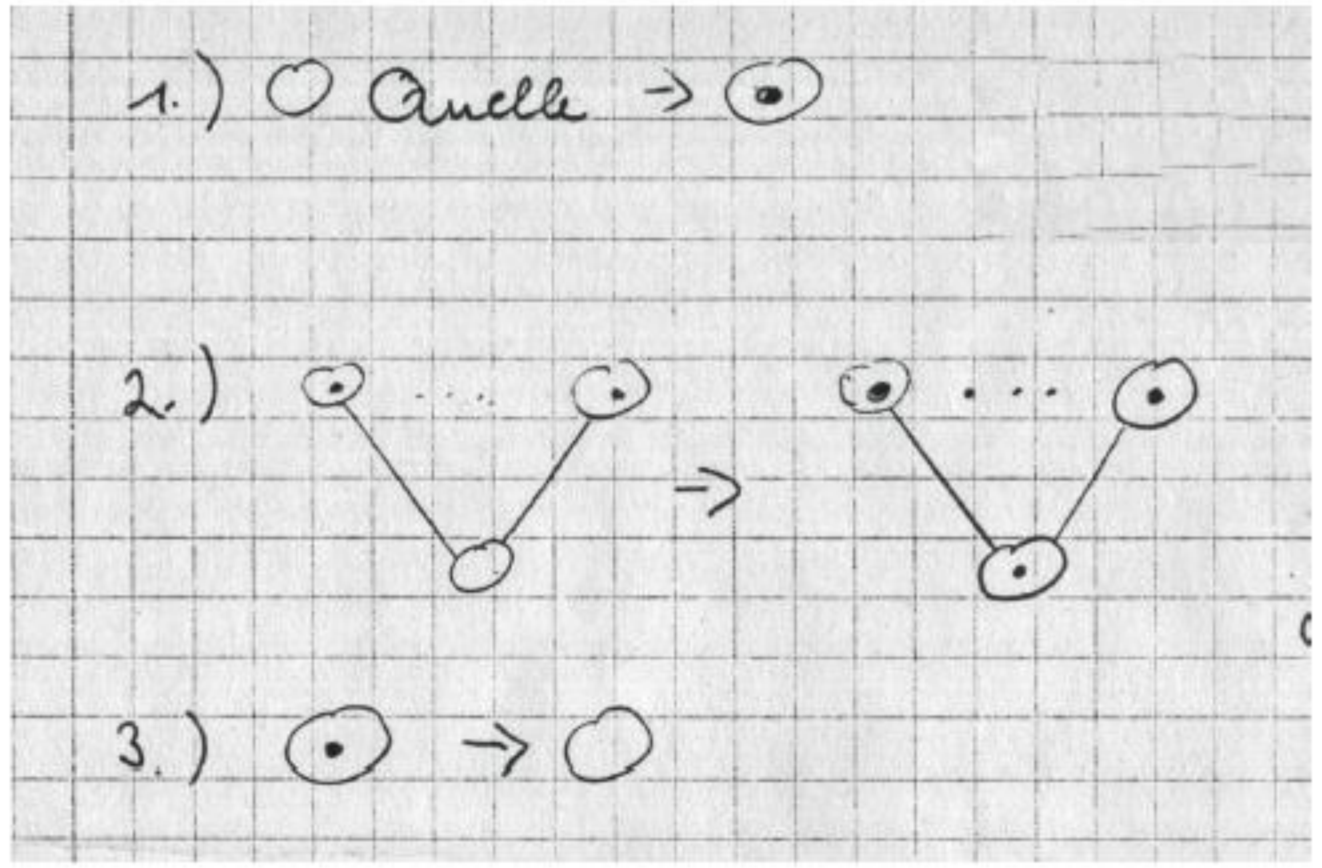
# 2 Pebble lemma

## 2.1 Trivial pebbling strategy

**Lemma 4.**

$$P_k(n) \leq 2n$$

- a node $u$ is isolated, when it is not connected to any edge; for each such node $u$ place a pebble on it and remove it in the next move.
- place pebbles on the end points $u$ of edges in the order of the depth of $u$. There are at most $|E|$ such nodes.




**def:** **indegree of a graph:** maximal number of direct ancestors of any node in the graph

**Lemma 3.** *Let $P_k(n)$ be the smallest number of pebbles sufficient to pebble any dag with n edges and indegree k. Then*

$$\begin{aligned} P_k(n) &= O(n/\log n) \\ &\leq C_k \cdot n/\log n \quad \textit{faa } n \end{aligned}$$

**Lemma 4.**

$$P_k(n) \leq 2n$$

## 2.2 Graph decompositions

- let $G = (V, E)$ be a directed acyclic graph
- graphs

$$G_1 = (V_1, E_1) \quad , \quad G_2 = (V_2, E_2)$$

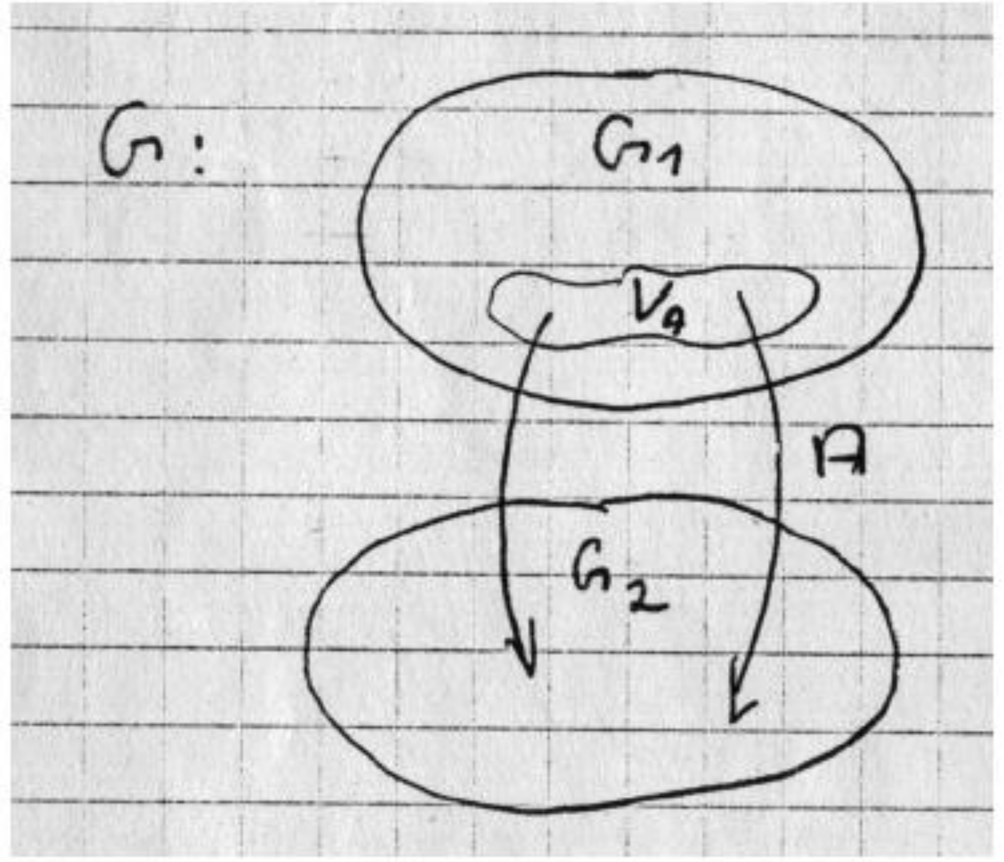
are a *decomposition* of $G$ if

1. $V_1$ and $V_2$ for a partition of $V$

$$V = V_1 \cup V_2 \quad , \quad V_1 \cap V_2 = \emptyset$$

2. for $i \in \{1, 2\}$ graph $G_i$ is the subgraph of $G$ spanned by $V_i$

$$E_i = E \cap (V_i \times V_i)$$



## 2.2 Graph decompositions

- let $G = (V, E)$ be a directed acyclic graph
- graphs

$$G_1 = (V_1, E_1) \quad , \quad G_2 = (V_2, E_2)$$

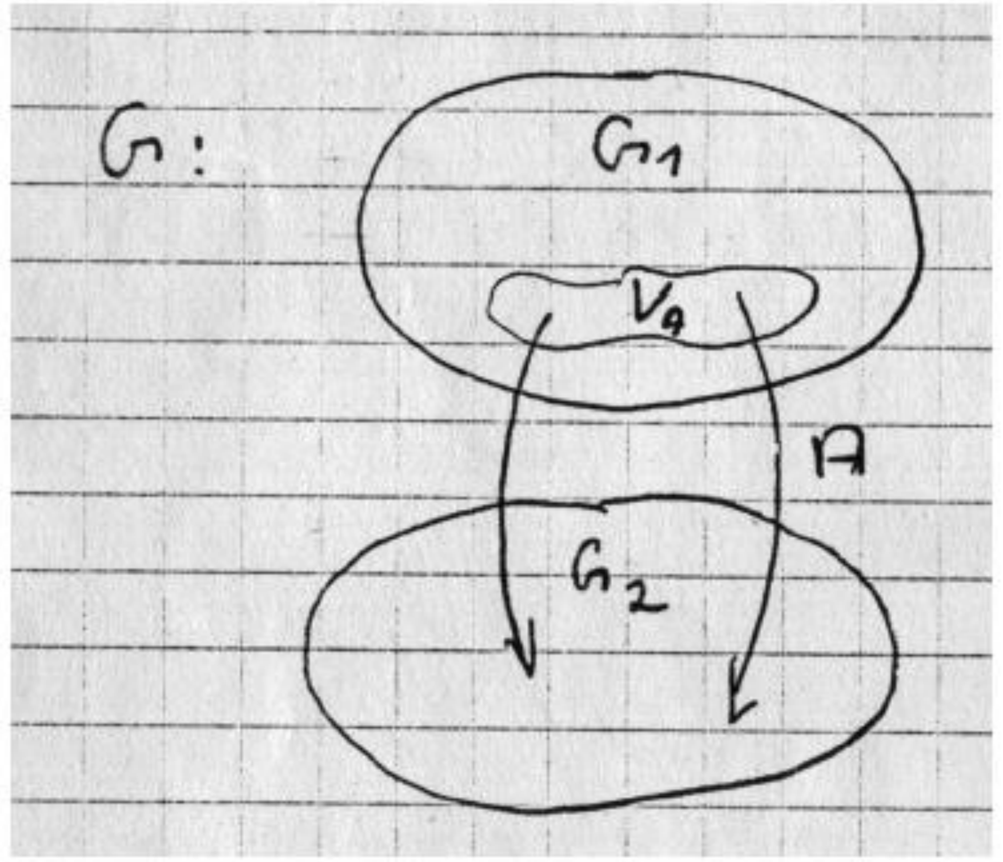
are a *decomposition* of $G$ if

1. $V_1$ and $V_2$ for a partition of $V$

$$V = V_1 \cup V_2 \quad , \quad V_1 \cap V_2 = \emptyset$$

2. for $i \in \{1, 2\}$ graph $G_i$ is the subgraph of $G$ spanned by $V_i$

$$E_i = E \cap (V_i \times V_i)$$



## 2.2 Graph decompositions

- let $G = (V,E)$ be a directed acyclic graph
- graphs

$$G_1 = (V_1, E_1) \quad , \quad G_2 = (V_2, E_2)$$

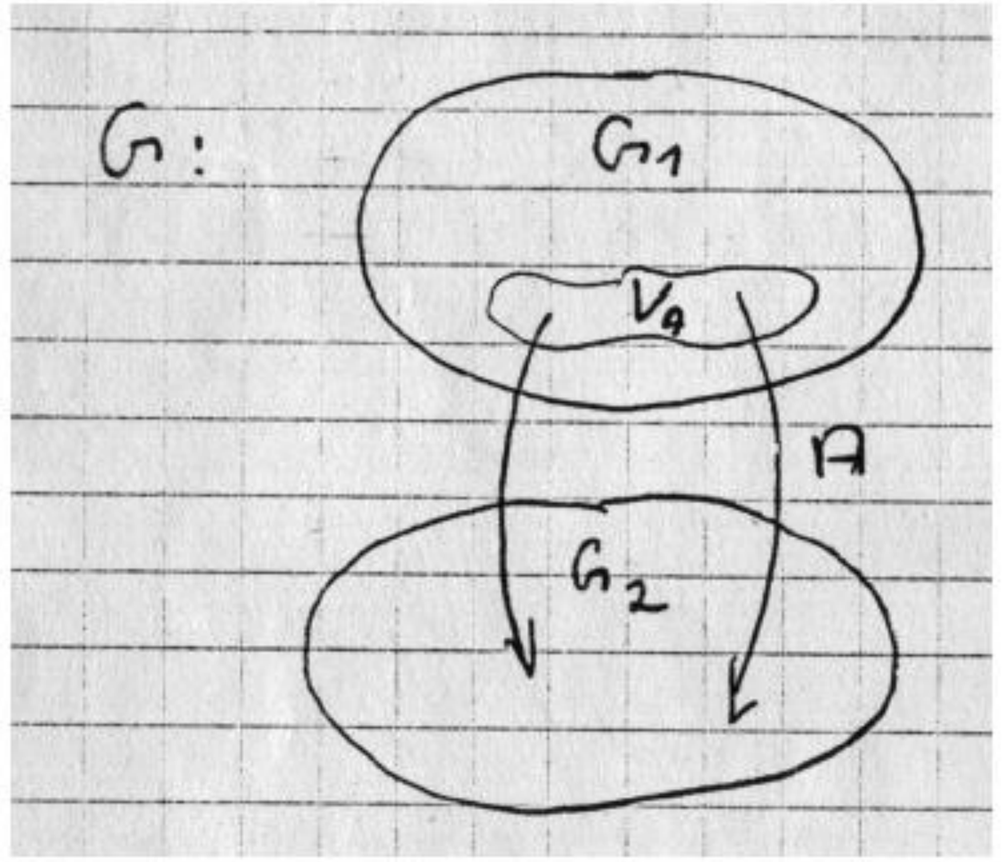
are a *decomposition* of $G$ if

1. $V_1$ and $V_2$ for a partition of $V$

$$V = V_1 \cup V_2 \quad , \quad V_1 \cap V_2 = \emptyset$$

2. for $i \in \{1,2\}$ graph $G_i$ is the subgraph of $G$ spanned by $V_i$

$$E_i = E \cap (V_i \times V_i)$$




- decomposition leaves remaining set edges going from $V_1$ to $V_2$

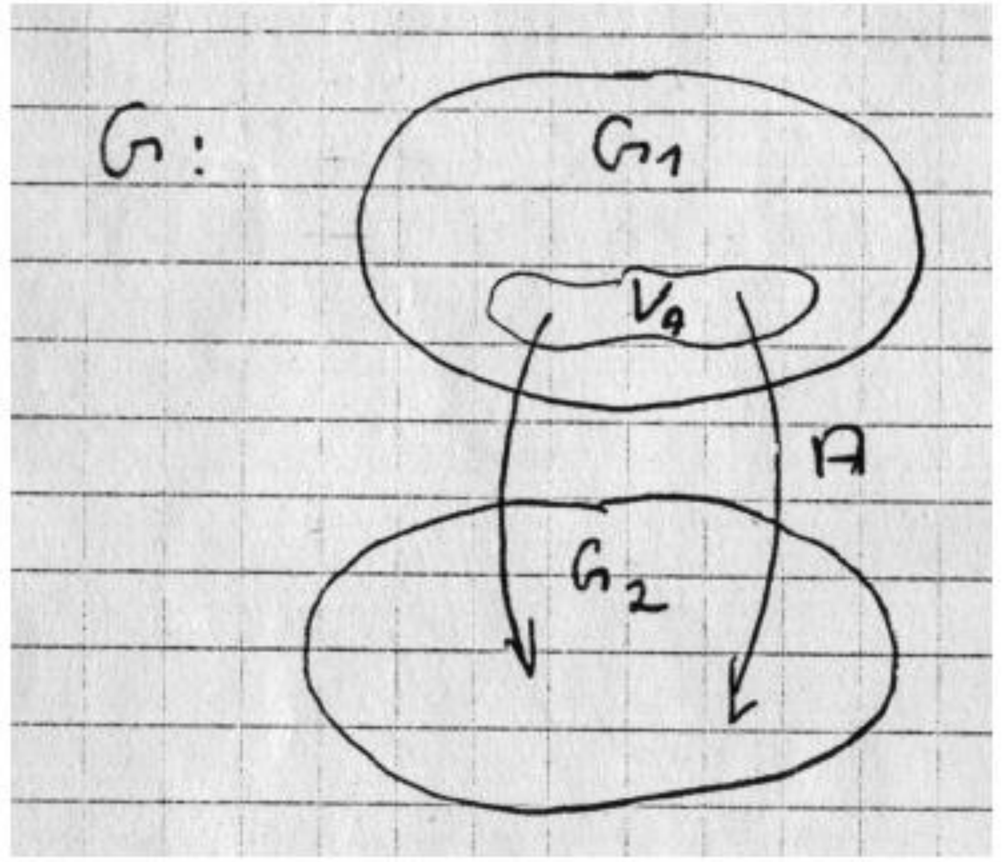
$$A = E \setminus (E_1 \cup E_2) \subseteq V_1 \times V_2$$

with set of start points

$$V_A = \{u \in V_1 \ : \ \exists v \in V_2 . \ (u,v) \in E\}$$

## 2.2 Graph decompositions

- let $G = (V, E)$ be a directed acyclic graph
- graphs
$$G_1 = (V_1, E_1) \quad , \quad G_2 = (V_2, E_2)$$
are a *decomposition* of $G$ if
1. $V_1$ and $V_2$ for a partition of $V$
$$V = V_1 \cup V_2 \quad , \quad V_1 \cap V_2 = \emptyset$$
2. for $i \in \{1,2\}$ graph $G_i$ is the subgraph of $G$ spanned by $V_i$
$$E_i = E \cap (V_i \times V_i)$$




- decomposition leaves remaining set edges going from $V_1$ to $V_2$
$$A = E \setminus (E_1 \cup E_2) \subseteq V_1 \times V_2$$
with set of start points
$$V_A = \{u \in V_1 \; : \; \exists v \in V_2. \; (u,v) \in E\}$$

**decomposing a graph into roughly equal size parts:**

**Lemma 5.** *Let $G = (V, E)$ be a dag with indegree $k$ and $n$ edges. Then $G$ can be decomposed into*
$$G_1 = (V_1, E_1) \quad , \quad G_2 = (V_2, E_2)$$
*such that*
$$n/2 \leq |E_1| < n/2 + k$$

## 2.2 Graph decompositions

- let $G = (V, E)$ be a directed acyclic graph
- graphs

$$G_1 = (V_1, E_1) \quad , \quad G_2 = (V_2, E_2)$$

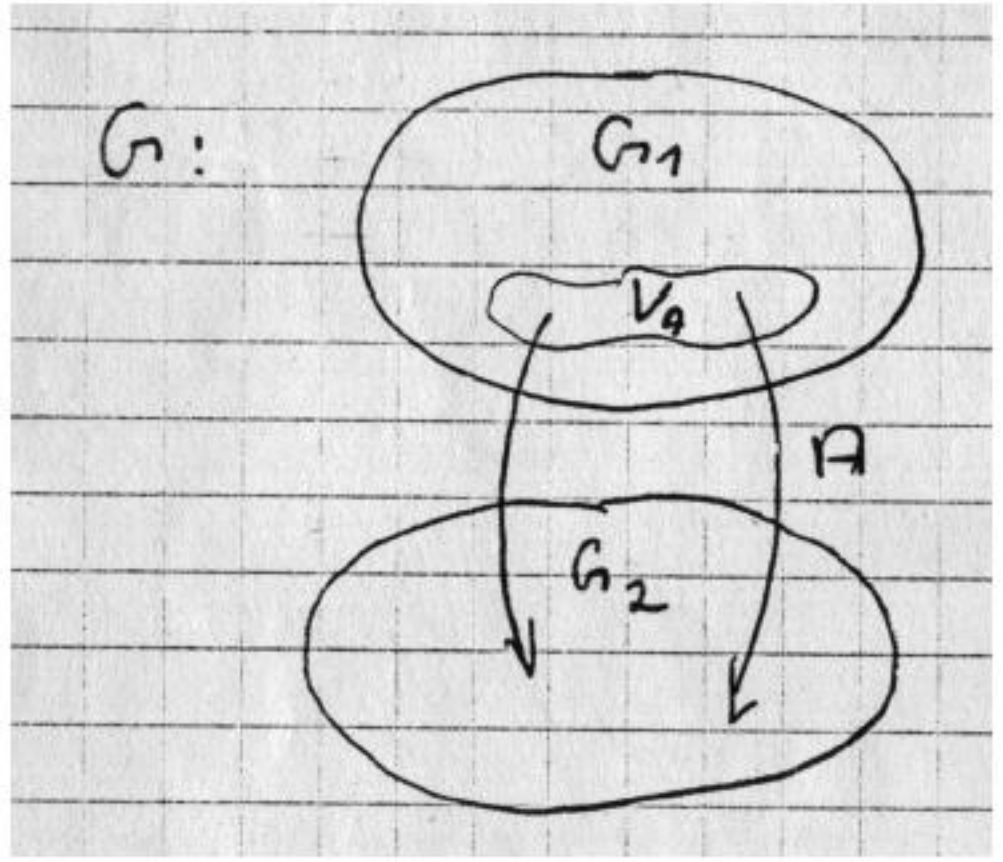
are a *decomposition* of $G$ if

1. $V_1$ and $V_2$ for a partition of $V$

$$V = V_1 \cup V_2 \quad , \quad V_1 \cap V_2 = \emptyset$$

2. for $i \in \{1,2\}$ graph $G_i$ is the subgraph of $G$ spanned by $V_i$

$$E_i = E \cap (V_i \times V_i)$$




- decomposition leaves remaining set edges going from $V_1$ to $V_2$

$$A = E \setminus (E_1 \cup E_2) \subseteq V_1 \times V_2$$

with set of start points

$$V_A = \{u \in V_1 \ : \ \exists v \in V_2. \ (u,v) \in E\}$$

**decomposing a graph into roughly equal size parts:**

**Lemma 5.** *Let $G = (V, E)$ be a dag with indegree $k$ and $n$ edges. Then $G$ can be decomposed into*

$$G_1 = (V_1, E_1) \quad , \quad G_2 = (V_2, E_2)$$

*such that*

$$n/2 \leq |E_1| < n/2 + k$$

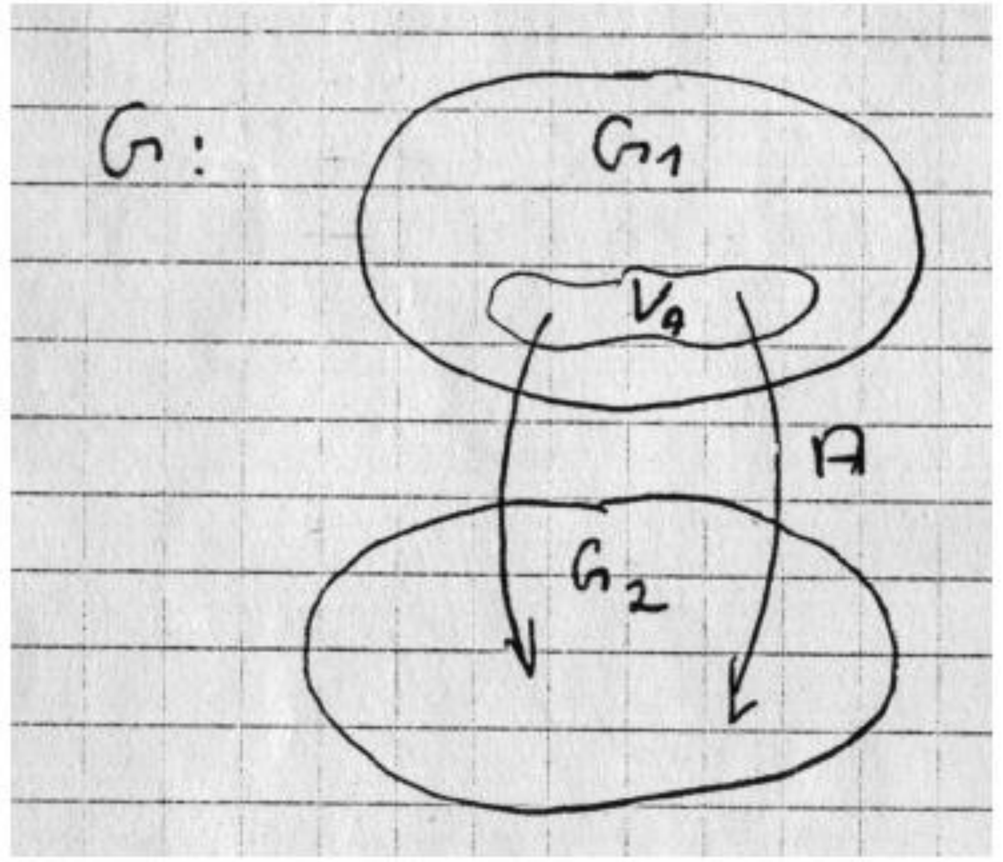
- w.l.o.g. graph $G$ has no isolated nodes (we can put them anywhere)
- start with

$$G_1 = G \quad , \quad G_2 = (\emptyset, \emptyset)$$

- move a sink $s$ of $G_1$ to $G_2$. This removes at most $k$ edges from $G_1$.
- Repeat until $|E_1| < n/2 + k$.

## 2.2 Graph decompositions

- let $G = (V, E)$ be a directed acyclic graph
- graphs
$$G_1 = (V_1, E_1) \quad , \quad G_2 = (V_2, E_2)$$
are a *decomposition* of $G$ if
  1. $V_1$ and $V_2$ for a partition of $V$
  $$V = V_1 \cup V_2 \quad , \quad V_1 \cap V_2 = \emptyset$$
  2. for $i \in \{1, 2\}$ graph $G_i$ is the subgraph of $G$ spanned by $V_i$
  $$E_i = E \cap (V_i \times V_i)$$




- decomposition leaves remaining set edges going from $V_1$ to $V_2$
$$A = E \setminus (E_1 \cup E_2) \subseteq V_1 \times V_2$$
with set of start points
$$V_A = \{u \in V_1 \ : \ \exists v \in V_2. \ (u, v) \in E\}$$

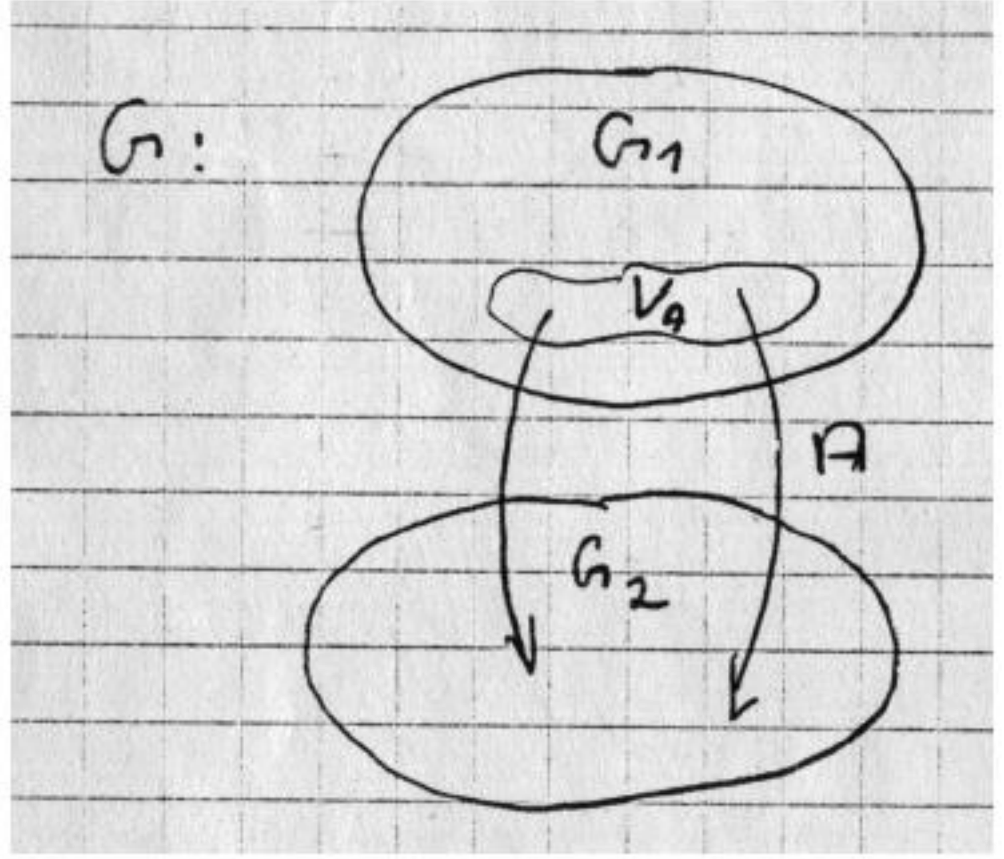
**decomposing a graph into roughly equal size parts:**

**Lemma 5.** *Let $G = (V, E)$ be a dag with indegree $k$ and $n$ edges. Then $G$ can be decomposed into*
$$G_1 = (V_1, E_1) \quad , \quad G_2 = (V_2, E_2)$$
*such that*
$$n/2 \leq |E_1| < n/2 + k$$

**def: fat and slim graphs** Decompose graph $G$ as above and consider the set $A$ of edges from $G_1$ to $G_2$ and its set $V_A$ of start points. Call the graph $G$

- *slim* if $|A| \leq 2n/\log n$
- *fat* if $|A| > 2n/\log n$

## 2.3 Recursive pebbling strategy




$n/2 \leq |E_1| < n/2 + k$

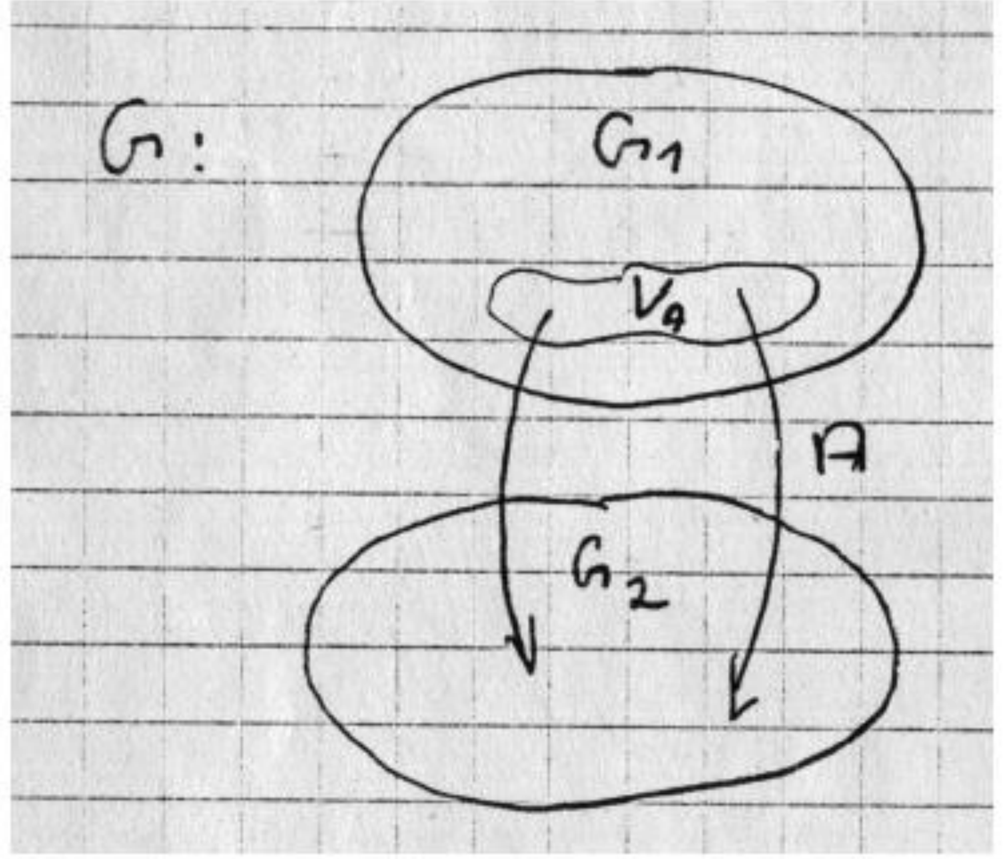
- *slim* if $|A| \leq 2n/\log n$
- *fat* if $|A| > 2n/\log n$

for a graph $G$ with $n$ edges

**base case** For $n \leq n_0$ (to be determined by the analysis) use the trivial pebbling strategy

$$n < n_0 : \quad P_k(n) \leq 2n$$

## 2.3 Recursive pebbling strategy




$n/2 \leq |E_1| < n/2 + k$

- *slim* if $|A| \leq 2n/\log n$
- *fat* if $|A| > 2n/\log n$

for a graph $G$ with $n$ edges

**base case** For $n \leq n_0$ (to be determined by the analysis) use the trivial pebbling strategy

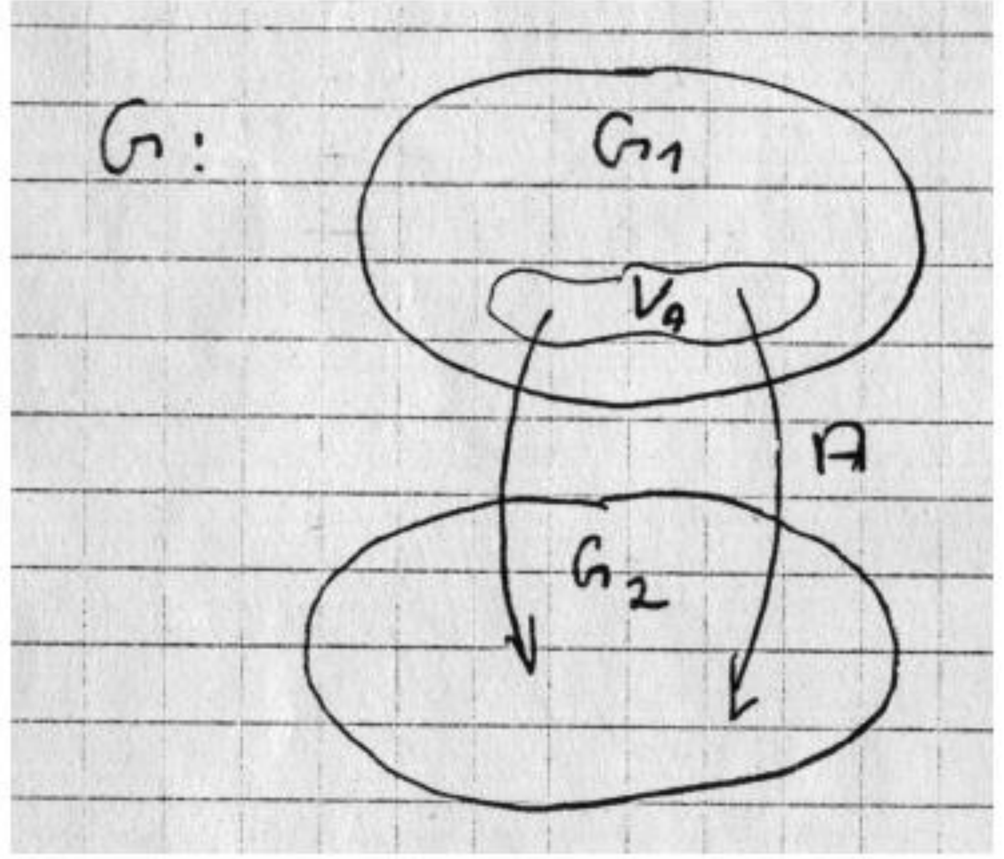
$$n < n_0 : \quad P_k(n) \leq 2n$$

For $n > n_0$ consider a balanced decomposition of $G$.

**case: $G$ is slim**

- pebble $G_1$ with $P_k(n/2+k)$ pebbles
- leave pebbles on $V_A$. This are at most $P_k(n/2+k)$ pebbles
- remove other pebbles from $G_1$
- pebble $G_2$ with $P_k(n/2) \leq P_k(n/2+k)$ pebbles

$$P_k(n) \leq P_k(n/2+k) + 2n/\log n$$




$$n/2 \le |E_1| < n/2 + k$$

- *slim* if $|A| \le 2n/\log n$
- *fat* if $|A| > 2n/\log n$

for a graph $G$ with $n$ edges

**base case** For $n \le n_0$ (to be determined by the analysis) use the trivial pebbling strategy

$$n < n_0 : \quad P_k(n) \le 2n$$

For $n > n_0$ consider a balanced decomposition of $G$.

**case: $G$ is slim**

$$P_k(n) \le P_k(n/2 + k) + 2n/\log n$$

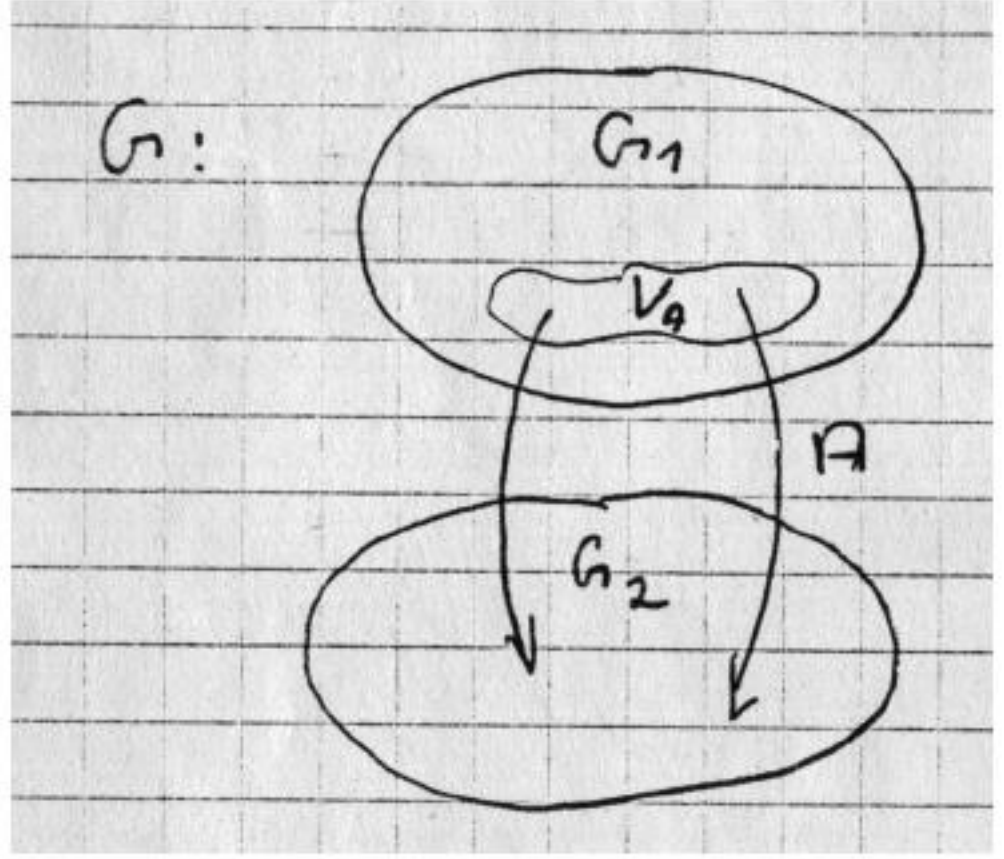
## 2.3 Recursive pebbling strategy

**case: $G$ is fat** Then

$$|E_2| = |E| - |E_1| - |A| \le n/2 - 2n/\log n$$

- start pebble strategy on $G_2$ using at most $P_k(n/2 - 2n/\log n)$ pebbles on $G_2$.

## 2.3 Recursive pebbling strategy




$$n/2 \le |E_1| < n/2 + k$$

- *slim* if $|A| \le 2n/\log n$
- *fat* if $|A| > 2n/\log n$

for a graph $G$ with $n$ edges

**base case** For $n \le n_0$ (to be determined by the analysis) use the trivial pebbling strategy

$$n < n_0 : \quad P_k(n) \le 2n$$

**case: $G$ is slim**

$$P_k(n) \le P_k(n/2+k) + 2n/\log n$$

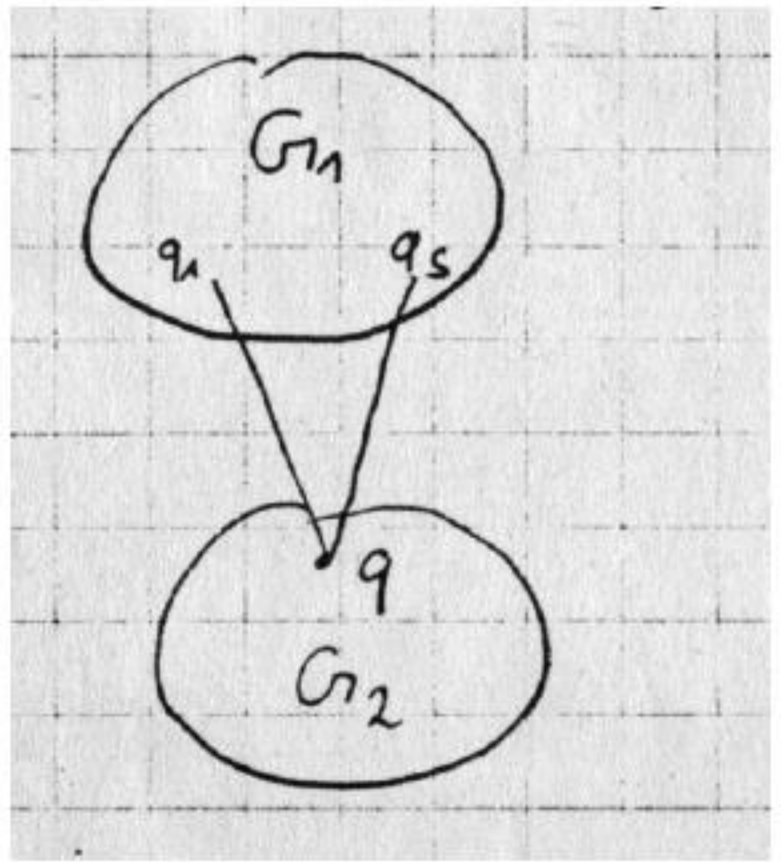
**case: $G$ is fat**

$$P_k(n) \le P_k(n/2+k) + P_k(n/2 - 2n/\log n) + k$$

**case: $G$ is fat** Then

$$|E_2| = |E| - |E_1| - |A| \le n/2 - 2n/\log n$$

- start pebble strategy on $G_2$ using at most $P_k(n/2 - 2n/\log n)$ pebbles on $G_2$.




- When a pebble must be placed on a source $q$ of $G$ interrupt strategy and
- using a pebble strategy for $G_1$ leave pebbles on the direct predecessors

$$q_1, \dots, q_s \in V_1$$

As $s \le k$ this uses at most $P_k(n/2+k) + k$ pebbles on $G_1$.

- place pebble on $q$, remove all pebbles on $G_1$ and continue the strategy on $G_2$

## 2.4 Solving the difference equation

for $n \le n_0$:

$$P_k(n) \le 2n$$

for $n > n_0$:

$$\begin{aligned} P_k(n) &\le \max\{P_k(n/2+k) + P_k(n/2 - 2n/\log n) + k, \\ &\quad P_k(n/2+k) + 2n/\log n\} \end{aligned}$$

## 2.4 Solving the difference equation

for $n \le n_0$:

$$P_k(n) \le 2n$$

for $n > n_0$:

$$\begin{aligned} P_k(n) &\le \max\{P_k(n/2+k) + P_k(n/2 - 2n/\log n) + k, \\ &\quad P_k(n/2+k) + 2n/\log n\} \end{aligned}$$

- Choose $n_0$ such that for all $n > n_0$ holds

$$\begin{aligned} \frac{n - 2k\log n}{2\log n(\log n - 2)} &\ge k && (1) \\ n &\ge 4k && (2) \\ n/2 - 2n/\log n &\ge n/4 && (3) \\ log(n/2+k) &\ge \frac{7}{8}\log n && (4) \end{aligned}$$

  and set

$$C = \max\{2\log n_0, 14\}$$

- show by induction on $n$

$$P_k(n) \le C \cdot n/\log n$$

## 2.4 Solving the difference equation

for $n \le n_0$:

$$P_k(n) \le 2n$$

for $n > n_0$:

$$\begin{aligned} P_k(n) \quad \le \quad & \max\{P_k(n/2+k)+P_k(n/2-2n/\log n)+k, \\ & P_k(n/2+k)+2n/\log n\} \end{aligned}$$

- Choose $n_0$ such that for all $n > n_0$ holds

$$\begin{aligned} \frac{n-2k\log n}{2\log n(\log n-2)} \quad &\ge \quad k && (1) \\ n \quad &\ge \quad 4k && (2) \\ n/2-2n/\log n \quad &\ge \quad n/4 && (3) \\ log(n/2+k) \quad &\ge \quad \frac{7}{8}\log n && (4) \end{aligned}$$

  and set

$$C = \max\{2\log n_0, 14\}$$

- show by induction on $n$

$$P_k(n) \le C \cdot n/\log n$$

**base case:** $n \le n_0$

$$P_k(n) \le 2n \le 2(\log n_0)n/\log n \le Cn/\log n$$

## 2.4 Solving the difference equation

for $n \le n_0$:

$$P_k(n) \le 2n$$

for $n > n_0$:

$$\begin{aligned} P_k(n) \quad \le \quad & \max\{P_k(n/2+k) + P_k(n/2 - 2n/\log n) + k, \\ & P_k(n/2+k) + 2n/\log n\} \end{aligned}$$

- Choose $n_0$ such that for all $n > n_0$ holds

$$\begin{aligned} \frac{n - 2k\log n}{2\log n(\log n - 2)} &\ge k && (1)\\ n &\ge 4k && (2)\\ n/2 - 2n/\log n &\ge n/4 && (3)\\ log(n/2+k) &\ge \frac{7}{8}\log n && (4) \end{aligned}$$

  and set

$$C = \max\{2\log n_0, 14\}$$

- show by induction on $n$

$$P_k(n) \le C \cdot n/\log n$$

**fat graphs:** first term in difference equation is maximum

$$\begin{aligned} P_k(n) &\le C\frac{n/2+k}{log(n/2+k)} + C\frac{n/2 - 2n/\log n}{\log(n/2 - 2n/\log n)} + k \\ &\le \frac{Cn/2}{\log n - 1} + C\frac{n/2 - 2n/\log n}{\log n - 2} + \frac{Ck}{\log n - 2} + k \quad \text{using (3)} \end{aligned}$$

## 2.4 Solving the difference equation

for $n \le n_0$:

$$P_k(n) \le 2n$$

for $n > n_0$:

$$\begin{aligned} P_k(n) \quad \le \quad & \max\{P_k(n/2+k) + P_k(n/2 - 2n/\log n) + k, \\ & P_k(n/2+k) + 2n/\log n\} \end{aligned}$$

- Choose $n_0$ such that for all $n > n_0$ holds

$$\begin{aligned} \frac{n - 2k\log n}{2\log n(\log n - 2)} &\ge k && (1) \\ n &\ge 4k && (2) \\ n/2 - 2n/\log n &\ge n/4 && (3) \\ \log(n/2+k) &\ge \frac{7}{8}\log n && (4) \end{aligned}$$

  and set

$$C = \max\{2\log n_0, 14\}$$

- show by induction on $n$

$$P_k(n) \le C \cdot n/\log n$$

**fat graphs:** first term in difference equation is maximum

$$\begin{aligned} P_k(n) \quad &\le \quad C\frac{n/2+k}{\log(n/2+k)} + C\frac{n/2 - 2n/\log n}{\log(n/2 - 2n/\log n)} + k \\ &\le \quad \frac{Cn/2}{\log n - 1} + C\frac{n/2 - 2n/\log n}{\log n - 2} + \frac{Ck}{\log n - 2} + k \quad \text{using (3)} \end{aligned}$$

apply

$$\frac{1}{x-a} = \frac{1}{x} + \frac{a}{x(x-a)} \text{ with } x = \log n, a \in \{1,2\}$$

to get

$$\begin{aligned} P_k(n) \quad \le \quad & \frac{Cn}{2\log n} + \frac{Cn}{2\log n(\log n - 1)} \\ & + \frac{Cn}{2\log n} + \frac{Cn}{\log n(\log n - 2)} \\ & - \frac{2Cn}{\log n(\log n - 2)} + \frac{Ck}{\log n - 2} + k \end{aligned}$$

## 2.4 Solving the difference equation

for $n \le n_0$:

$$P_k(n) \le 2n$$

for $n > n_0$:

$$\begin{aligned} P_k(n) \quad \le \quad & \max\{P_k(n/2+k) + P_k(n/2 - 2n/\log n) + k, \\ & P_k(n/2+k) + 2n/\log n\} \end{aligned}$$

- Choose $n_0$ such that for all $n > n_0$ holds

$$\begin{aligned} \frac{n - 2k\log n}{2\log n(\log n - 2)} &\ge k && (1) \\ n &\ge 4k && (2) \\ n/2 - 2n/\log n &\ge n/4 && (3) \\ log(n/2+k) &\ge \frac{7}{8}\log n && (4) \end{aligned}$$

  and set

$$C = \max\{2\log n_0, 14\}$$

- show by induction on $n$

$$P_k(n) \le C \cdot n/\log n$$

**fat graphs:** first term in difference equation is maximum

$$\begin{aligned} P_k(n) &\le C\frac{n/2+k}{log(n/2+k)} + C\frac{n/2 - 2n/\log n}{\log(n/2 - 2n/\log n)} + k \\ &\le \frac{Cn/2}{\log n - 1} + C\frac{n/2 - 2n/\log n}{\log n - 2} + \frac{Ck}{\log n - 2} + k \quad \text{using (3)} \end{aligned}$$

apply

$$\frac{1}{x-a} = \frac{1}{x} + \frac{a}{x(x-a)} \text{ with } x = \log n, a \in \{1,2\}$$

to get

$$\begin{aligned} P_k(n) \le\ & \frac{Cn}{2\log n} + \frac{Cn}{2\log n(\log n - 1)} \\ & + \frac{Cn}{2\log n} + \frac{Cn}{\log n(\log n - 2)} \\ & - \frac{2Cn}{\log n(\log n - 2)} + \frac{Ck}{\log n - 2} + k \end{aligned}$$

$$\begin{aligned} &\le \frac{Cn}{\log n} - \frac{Cn}{2\log n(\log n - 2)} + \frac{Ck}{\log n - 2} + k \\ &\le \frac{Cn}{\log n} \quad \text{using (1)} \end{aligned}$$

## 2.4 Solving the difference equation

for $n \le n_0$:

$$P_k(n) \le 2n$$

for $n > n_0$:

$$\begin{aligned} P_k(n) \quad &\le \quad \max\{P_k(n/2+k) + P_k(n/2 - 2n/\log n) + k, \\ &\qquad P_k(n/2+k) + 2n/\log n\} \end{aligned}$$

- Choose $n_0$ such that for all $n > n_0$ holds

$$\begin{aligned} \frac{n - 2k\log n}{2\log n(\log n - 2)} &\ge k && (1) \\ n &\ge 4k && (2) \\ n/2 - 2n/\log n &\ge n/4 && (3) \\ log(n/2+k) &\ge \frac{7}{8}\log n && (4) \end{aligned}$$

  and set

$$C = \max\{2\log n_0, 14\}$$

- show by induction on $n$

$$P_k(n) \le C \cdot n/\log n$$

**slim graphs:** second term in difference equation is maximum

$$\begin{aligned} P_k(n) &\le C\frac{n/2+k}{\log(n/2+k)} + \frac{2n}{\log n} \\ &\le \frac{3Cn}{4\log(n/2+k)} + \frac{2n}{\log n} \quad \text{using (2)} \end{aligned}$$

## 2.4 Solving the difference equation

for $n \le n_0$:

$$P_k(n) \le 2n$$

for $n > n_0$:

$$\begin{aligned} P_k(n) \quad \le \quad & \max\{P_k(n/2+k) + P_k(n/2 - 2n/\log n) + k, \\ & P_k(n/2+k) + 2n/\log n\} \end{aligned}$$

- Choose $n_0$ such that for all $n > n_0$ holds

$$\begin{aligned} \frac{n - 2k\log n}{2\log n(\log n - 2)} &\ge k && (1)\\ n &\ge 4k && (2)\\ n/2 - 2n/\log n &\ge n/4 && (3)\\ log(n/2+k) &\ge \frac{7}{8}\log n && (4) \end{aligned}$$

and set

$$C = \max\{2\log n_0, 14\}$$

- show by induction on $n$

$$P_k(n) \le C \cdot n/\log n$$

**slim graphs:** second term in difference equation is maximum

$$\begin{aligned} P_k(n) &\le C\frac{n/2+k}{\log(n/2+k)} + \frac{2n}{\log n} \\ &\le \frac{3Cn}{4\log(n/2+k)} + \frac{2n}{\log n} \quad \text{using (2)} \\ &\le \frac{6}{7}\frac{Cn}{\log n} + \frac{2n}{\log n} \quad \text{using (4)} \\ &\le Cn/\log n \quad (C \ge 14) \end{aligned}$$

# 3 TM computation graphs

## 3.1 Defining the graphs

- Consider $k$-tape TM $M$, input $w$ with length $|w| = n$ and number $t \geq n$ of steps. Let

$$\lambda = \lceil t^{2/3} \rceil$$

- divide

  1. time into *time intervals* of $\lambda$ steps
  2. tapes into *blocks* of length $\lambda$

- number of time intervals

$$\tau = \lceil t/\lambda \rceil + 1 = O(t^{1/3})$$

# 3 TM computation graphs

## 3.1 Defining the graphs

- Consider $k$-tape TM $M$, input $w$ with length $|w| = n$ and number $t \geq n$ of steps. Let
$$\lambda = \lceil t^{2/3} \rceil$$
- divide
  1. time into *time intervals* of $\lambda$ steps
  2. tapes into *blocks* of length $\lambda$
- number of time intervals
$$\tau = \lceil t/\lambda \rceil + 1 = O(t^{1/3})$$
- for time intervals $i \in [0 : \tau]$ define the *result*
$$res(i)$$
  of the time interval as
  1. state and head positions
  2. inscriptions of the blocks visited in the time interval (at most $2k$)

  at the end of the time interval. For $i = 0$ take initial state and tape inscription.
- length
$$|res(i)| = O(\lambda) = O(t^{2/3})$$

### 3.1 Defining the graphs

- Consider $k$-tape TM $M$, input $w$ with length $|w| = n$ and number $t \geq n$ of steps. Let
$$\lambda = \lceil t^{2/3} \rceil$$
- divide
  1. time into *time intervals* of $\lambda$ steps
  2. tapes into *blocks* of length $\lambda$
- number of time intervals
$$\boxed{\tau = \lceil t/\lambda \rceil + 1 = O(t^{1/3})}$$
- for time intervals $i \in [0:\tau]$ define the *result*
$$res(i)$$
  of the time interval as
  1. state and head positions
  2. inscriptions of the blocks visited in the time interval (at most $2k$)

  at the end of the time interval. For $i = 0$ take initial state and tape inscription.
- length
$$\boxed{|res(i)| = O(\lambda) = O(t^{2/3})}$$

**computation graph**
$$G = (V,E)$$

- nodes: the time intervals
$$V = [0:\tau]$$
- edge
$$(i,j) \in E$$
  if $res(i)$ is needed to simulate time interval $j$, i.e. if

1. $j = i+1$ for state at the start of interval $j$ or
2. there is a block visited in intervals $i$ and $j$ but not in between or
3. $i = 0$ and there is a block visited for the first time in time interval $j$

$\boxed{\textbf{indegree:}\quad 2k+1}$

**simulation of time interval $j$:**

- possible if $res(i)$ is available for all direct predecessors of $i$ in $G$.
- rule of the pebble game with pebbles $res(x)$.
- whenever pebble is placed on node $j$ compute $res(j)$
- whenever pebble is removed from node $i$ erase $res(i)$
- space used is number of pebbles times pebble size
$$\boxed{O(p \cdot \lambda) = O((t^{1/3}/\log t) \cdot t^{2/3}) = O(t/\log t)}$$

## 3.3 Implementation details

- writing down $G$: space

$$O(\tau \cdot \log \tau) = O(t^{1/3} \log t)$$

### 3.3 Implementation details

- writing down $G$: space

$$O(\tau \cdot \log \tau) = O(t^{1/3} \log t)$$

- finding $G$:

  1. brute force: enumerate all graphs in some order. Try simulation for each graph

  2. or construct graphs on the fly. Once the number of edges in the graph exceeds $n_0$ from the pebble lemma, switch to recursive pebbling strategy

### 3.3 Implementation details

- writing down $G$: space

$$O(\tau \cdot \log \tau) = O(t^{1/3} \log t)$$

- finding $G$:

  1. brute force: enumerate all graphs in some order. Try simulation for each graph

  2. or construct graphs on the fly. Once the number of edges in the graph exceeds $n_0$ from the pebble lemma, switch to recursive pebbling strategy

- implementing the recursion

  1. parameter for 1 recursive call: a subgraph and some nodes, where to leave pebbles. Space
  $$O(\tau \cdot \log \tau)$$

  2. recursion depth $\leq \tau$

  Total space for implementation of recursion

$$O(\tau^2 \log \tau) = O(t^{2/3} \log t)$$

### 3.3 Implementation details

- writing down $G$: space

$$O(\tau \cdot \log \tau) = O(t^{1/3} \log t)$$

- finding $G$:

  1. brute force: enumerate all graphs in some order. Try simulation for each graph

  2. or construct graphs on the fly. Once the number of edges in the graph exceeds $n_0$ from the pebble lemma, switch to recursive pebbling strategy

- implementing the recursion

  1. parameter for 1 recursive call: a subgraph and some nodes, where to leave pebbles. Space
$$O(\tau \cdot \log \tau)$$

  2. recursion depth $\leq \tau$

  Total space for implementation of recursion

$$O(\tau^2 \log \tau) = O(t^{2/3} \log t)$$

- simulating $t(n)$ steps when $t$ is not space constructible: try simulation for

$$t = n, 2n, 4n, \ldots$$