Lecture 14: DP III

Seen so far:
- Fibo
- String in L^* (L^k)
- How to come up with DP
  - Solve recursively (memoization)
  - Figure out table + filling order)
  - DP!
- Edit distance
- CYK.

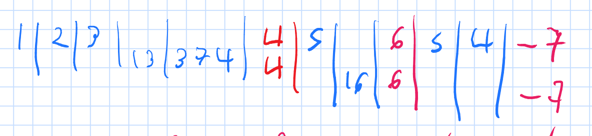
Today

- Longest common subsequence.
- Max independent set in a tree.
- DAGs: Computing the DP order directly [longest path in a DAG.]

Longest common subsequence

$\alpha_1, \alpha_2, \dots, \alpha_n$
$\beta_1, \beta_2, \dots, \beta_n$

1, 2, 3, 4, 5, 6, 5, 4 ~ 7   3
13, 374, 4, 16, 6, ~ 7



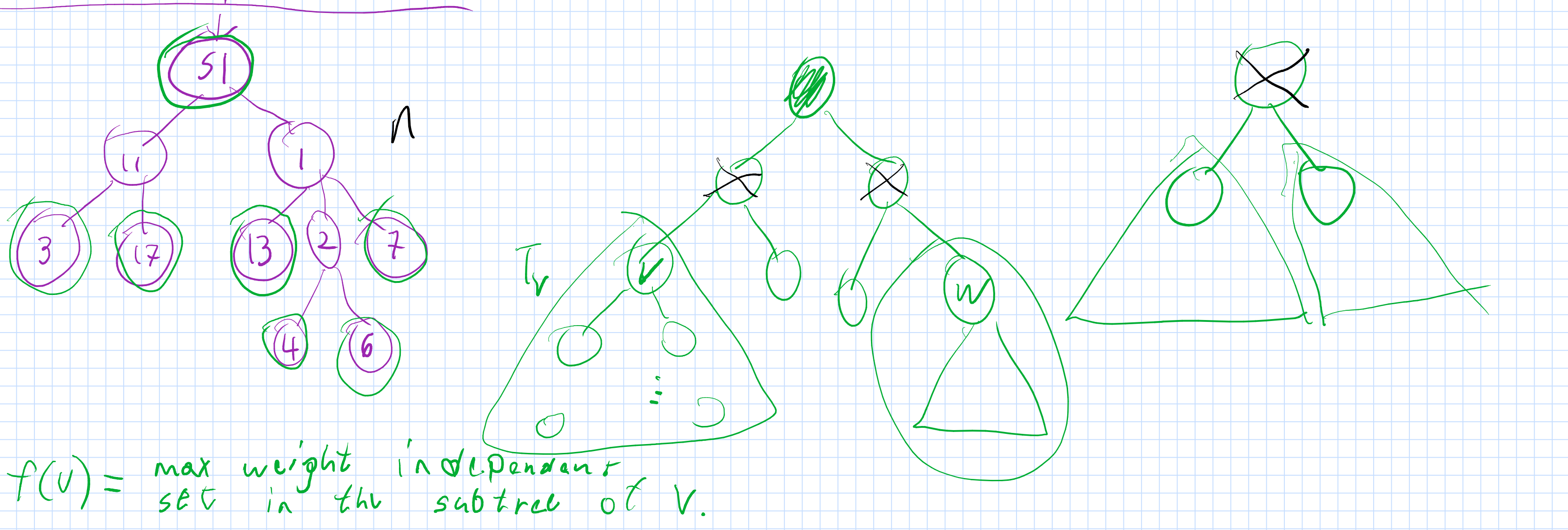
$f(i,j)$ = len of longest subseq of $\alpha_1, \dots, \alpha_i$ and $\beta_1, \dots, \beta_j$.

$$f(i,j) = \begin{cases} 0 & i=0 \text{ or } j=0 \\ \max\begin{cases} f(i-1,j) \\ f(i,j-1) \\ f(i-1,j-1)+1 & \alpha_i = \beta_j \end{cases} \end{cases}$$

$O(nm)$

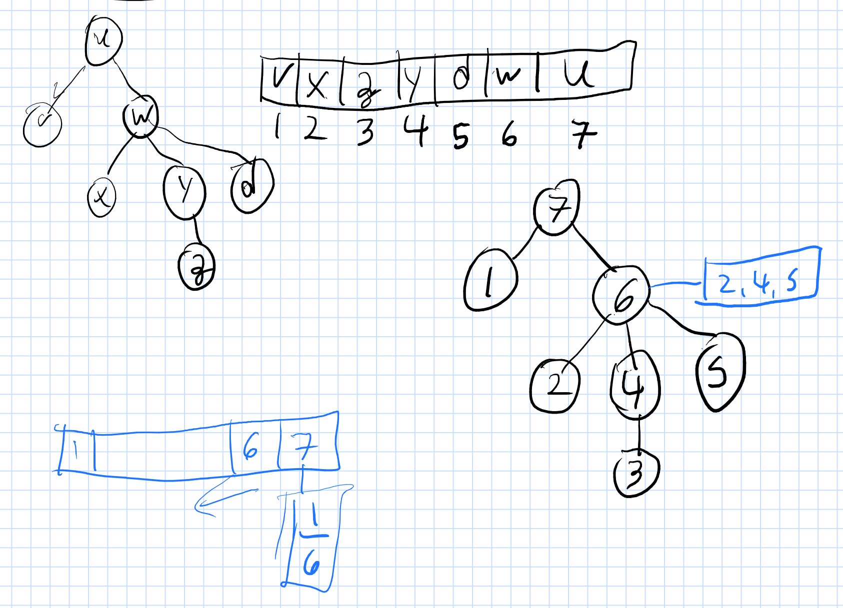
LIS: $O(n \log n)$ time using data-structures

Max W. Indep set tree

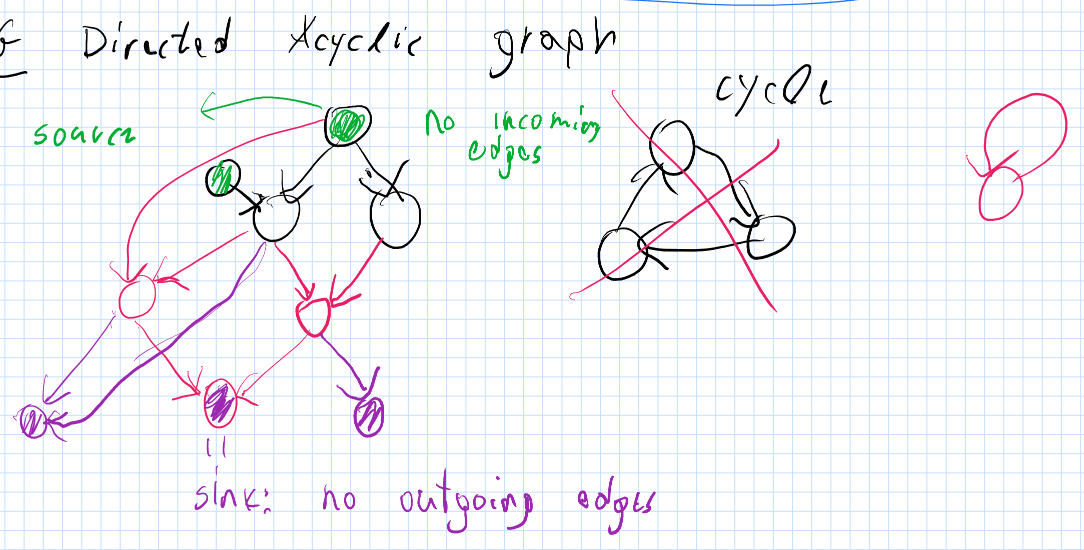


$f(v)$ = max weight independent set in the subtree of v.

$$f(v) = \begin{cases} \text{weight}(v) & v \text{ has no children} \\ \max\begin{cases} \text{weight}(v) + \sum_{u \in \text{grandchild}(v)} f(u) \\ \sum_{u \in \text{child}(v)} f(u) \end{cases} \end{cases}$$

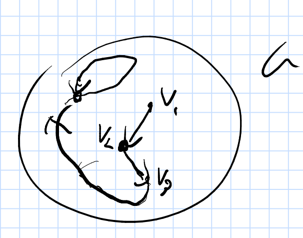


DAG Directed Acyclic graph



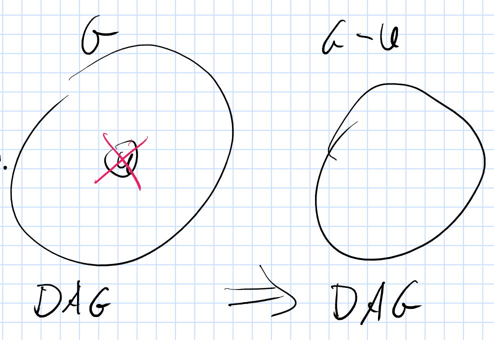
Claim In a DAG there is always at least one source, and at least one sink.

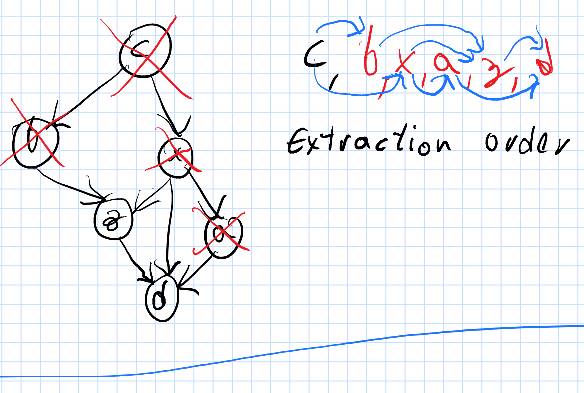
Proof
G be a DAG with n vertices and m edges.



v_1 = arbitrary vertex
Let $i=1$, if v_i is a sink — done.
Otherwise, let v_{i+1} be any neighbor of v_i s.t. $v_i \to v_{i+1}$ is an edge outgoing from v_i. $v_1 \to v_2 \to v_3 \to \dots \to v_i$
(v_i, v_{i+1})
$(v_i \to v_{i+1})$
v_i. Set $i \leftarrow i+1$, and repeat the process.
Since the process can not visit the same vertex twice (because then we have a cycle in the graph), and the graph is finite, the process must stop at a sink. ■

Observation
DAG is still a DAG after vertex deletion.





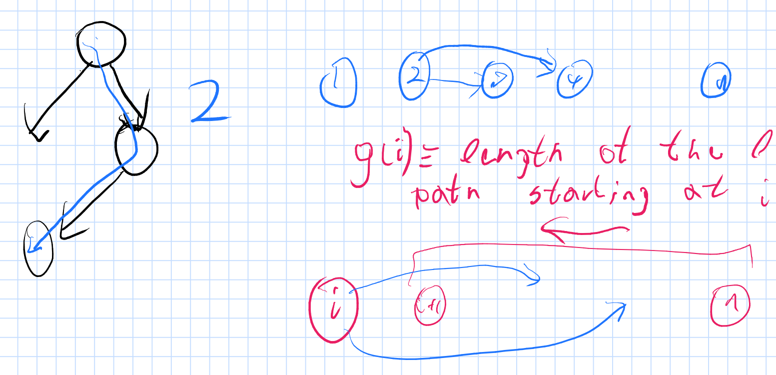
c, b, x, a, z, d
Extraction order

Claim One can compute the extraction order of a DAG in $O(n+m)$ time.

$1, 2, \dots, n$
$e_1, \dots, e_m$ $e_i = (s_i, t_i) = s_i \to t_i$

"Extraction order" ≡ Topological ordering

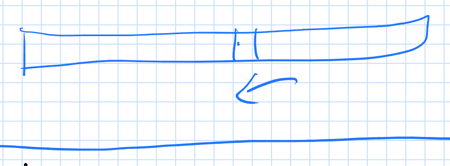
G: DAG
Compute the longest path in G.



$g(i)$ ≡ length of the longest path starting at i

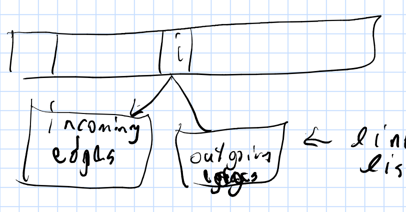
$$g(i) = \begin{cases} 0 & \text{if } i \text{ has no children} \\ 1 + \max_{i \to k \in E(G)} g(k) \end{cases}$$

$O(n+m)$



G undirected graph
⟹ Turn into a DAG

G
$1, 2, \dots, n$
$e_1, e_2, \dots, e_m$



$i \to j$

$O(n+m)$ $\forall v$ in 1:n $d_{in}(v)$ $d_{out}(v)$ $O(n+m)$

W = list of all current sources.

w