Algorithm Design & Analysis (CSE222)

Lecture-8

Recap

- Fibonacci numbers

Recap

- Fibonacci numbers

M = {}

```
Fib(n)
if n == 1
        return 0
if n == 2
        return 1
if M[n] is not empty
        return M[n]
else
        M[n] = Fib(n − 1) + Fib(n − 2)
return M[n]
```

Recap

- Fibonacci numbers

- Collection of non-adjacent heavy set
    - Greedy approach need not be optimal

1, 5, 8, 5, 4, 7

Outline

- Collection of non-adjacent heavy set
    - Dynamic Programming

- Vertex Cover

Collection of non-adjacent heavy set

<u>Subproblem:</u>

MaxWt[k] = The weight of an optimal set $B^* \subseteq \{b_1, b_2, \ldots, b_k\}$ such that no two balls are adjacent to each other and their sum of weights is maximum.

$$B^* = \{\underbrace{b_{i_1}, b_{i_2}, \ldots}_{\text{OPT for } \{b_1, b_2 \ldots b_{k-2}\}}, \cancel{b_k}\}$$

If k = 0 then MaxWt[0] = 0.
If k = 1 then MaxWt[0] = b_1.

Claim: Let B^* be a collection of non-adjacent balls and their sum is maximum. We call B^* be the optimal solution for the set $\{b_1, b_2, \ldots, b_k\}$.

1. If $b_k \in B^*$, then $B^* \setminus \{b_k\}$ is an optimal solution for $\{b_1, b_2, \ldots, b_{k-2}\}$.
2. If $b_k \notin B^*$, then B^* is an optimal solution for $\{b_1, b_2, \ldots, b_{k-1}\}$.

Proof by contradiction

- Identify the proposition (P) that has to be proved.
- Assume P is false, i.e., ~P is true.
- Check if, ~P being true implies falsehood, i.e., two mutually contradictory assertions (e.g., Q and ~Q both are true).
- Since, P being false leads to contradiction, hence P has to be true.

Proof:

$p \to q$

$(\sim p \vee q)$ $\qquad$ $p \cap \sim q$

Claim: Let B* be a collection of non-adjacent balls and their sum is maximum. We call B* be the optimal solution for the set $\{b_1, b_2, ..., b_k\}$.

1. If $\underbrace{b_k \in B^*}_{p}$, then $\underbrace{B^* \setminus \{b_k\} \text{ is an optimal solution for } \{b_1, b_2, ..., b_{k-2}\}}_{q}$.

<u>Proof:</u> If $b_k \in B^*$ then $b_{k-1} \notin B^*$.

$B^* \setminus \{b_k\}$ is not opt. of $\{b_1, b_2, \ldots b_{k-2}\}$

C is opt of $(b_1, b_2, \ldots, b_{k-2})$

Proof:

Claim: Let B* be a collection of non-adjacent balls and their sum is maximum. We call B* be the optimal solution for the set $\{b_1, b_2, \ldots, b_k\}$.

1. If $b_k \in B^*$, then $B^* \setminus \{b_k\}$ is an optimal solution for $\{b_1, b_2, \ldots, b_{k-2}\}$.

<u>Proof:</u> If $b_k \in B^*$ then $b_{k-1} \notin B^*$. Suppose, $B^* \setminus \{b_k\}$ is not an optimal solution for the set $\{b_1, b_2, \ldots, b_{k-2}\}$. Let, $C \subseteq \{b_1, b_2, \ldots, b_{k-2}\}$ be optimal solution for the set.

$$W(C) = \sum_{b \in C} w(b) > \sum_{b \in B^* \setminus \{b_k\}} w(b)$$

$C \cup \{b_k\}$ vs B^*

$$W(C \cup \{b_k\}) = \underline{\sum_{b \in C} w(b)} + \underline{w(b_k)} \boxed{>} \underline{\sum_{b \in B^* \setminus \{b_k\}} w(b)} + \underline{w(b_k)}$$

Proof:

Claim: Let B* be a collection of non-adjacent balls and their sum is maximum. We call B* be the optimal solution for the set $\{b_1, b_2, \ldots, b_k\}$.

1. If $b_k \in B^*$, then $B^* \setminus \{b_k\}$ is an optimal solution for $\{b_1, b_2, \ldots, b_{k-2}\}$.

<u>Proof:</u> If $b_k \in B^*$ then $b_{k-1} \notin B^*$. Suppose, $B^* \setminus \{b_k\}$ is not an optimal solution for the set $\{b_1, b_2, \ldots, b_{k-2}\}$. Let, $C \subseteq \{b_1, b_2, \ldots, b_{k-2}\}$ be optimal solution for the set.

Compare $W(C)$ with $W(B^* \setminus \{b_k\})$

Proof:

Claim: Let B* be a collection of non-adjacent balls and their sum is maximum. We call B* be the optimal solution for the set $\{b_1, b_2, \ldots, b_k\}$.

1. If $b_k \in B^*$, then $B^* \setminus \{b_k\}$ is an optimal solution for $\{b_1, b_2, \ldots, b_{k-2}\}$.
2. If $b_k \notin B^*$, then B* is an optimal solution for $\{b_1, b_2, \ldots, b_{k-1}\}$.

$\{b_1, b_2 \ldots b_k\}$

(B^*)

$B^* \subseteq B_k$

$B^* \subseteq B_{k-1}$

Proof:

Claim: Let B* be a collection of non-adjacent balls and their sum is maximum. We call B* be the optimal solution for the set $\{b_1, b_2, \ldots, b_k\}$.

1. If $b_k \in B^*$, then $B^* \setminus \{b_k\}$ is an optimal solution for $\{b_1, b_2, \ldots, b_{k-2}\}$.
2. If $b_k \notin B^*$, then B* is an optimal solution for $\{b_1, b_2, \ldots, b_{k-1}\}$.

Proof: If $b_k \notin B^*$ then $B^* \subseteq \{b_1, b_2, \ldots, b_{k-1}\}$. Suppose, $C \subseteq \{b_1, b_2, \ldots, b_{k-1}\}$ be an optimal solution set.

Compare W(C) with W(B*)

$w(k) =$

Recurrence

For k > 1, $\mathrm{MaxWt}[k] = \max\{\mathrm{MaxWt}[k-1], w(k) + \mathrm{MaxWt}[k-2]\}$

b_1 b_2 b_3 b_4 b_5 b_6

Example, 1, 5, 8, 5, 4, 7

$M(0) = 0$, $M(1) = 1$

$M(2) = \max(1, 5+0) = 5$

$M(3) = \max(5, 8+1) = 9$

$M(4) = \max(9, 5+5) = 10$

Algorithm

Input: Weights of Balls
Output: Maximum Weight of non-adjacent set.

Running time: O(|W|)

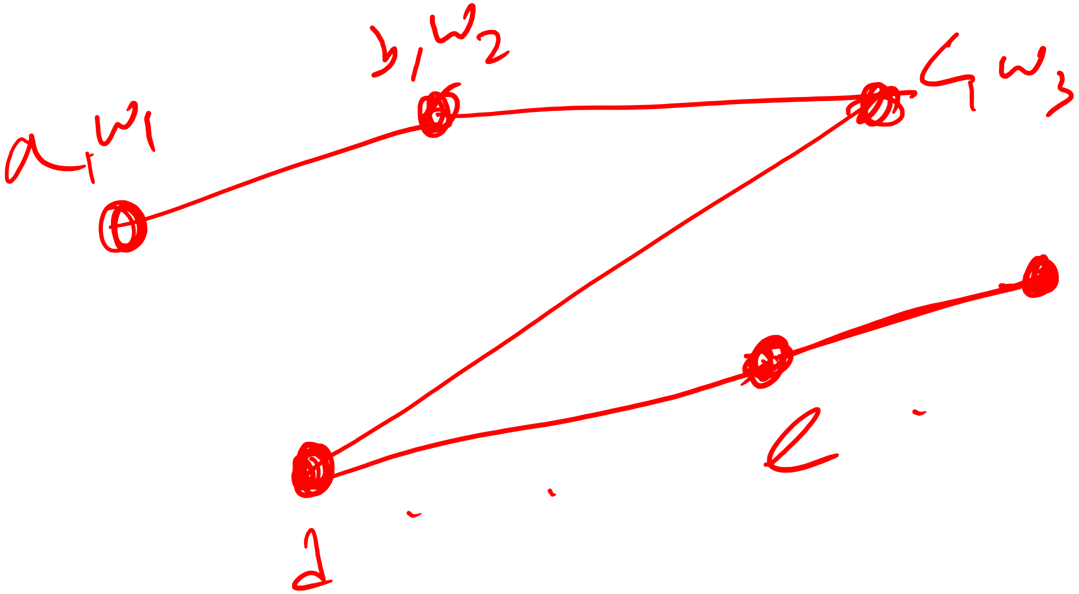
$\textsc{MaximumWeight}(W)$

$\mathrm{MaxWt}[0] = 0;\ \mathrm{MaxWt}[1] = W[1];$
for $i = 2 \cdots |W|$
    $t_1 = W[i] + \mathrm{MaxWt}[i-2];$
    $t_2 = \mathrm{MaxWt}[i-1];$
    if $(t_1 > t_2)$
        $\mathrm{MaxWt}[i] = t_1;$
    else
        $\mathrm{MaxWt}[i] = t_2;$
return $\mathrm{MaxWt}[|W|]$

Independent Set

For a given graph G = (V,E), a set of vertices S ⊆ V(G) is independent if for every pair u, v ∈ S, (u, v) ∉ E(G).

Question: How can we efficiently compute the maximum weighted independent set in a path graph?



Outline

- Collection of non-adjacent heavy set
  - Dynamic Programming

- Vertex Cover

Vertex Cover

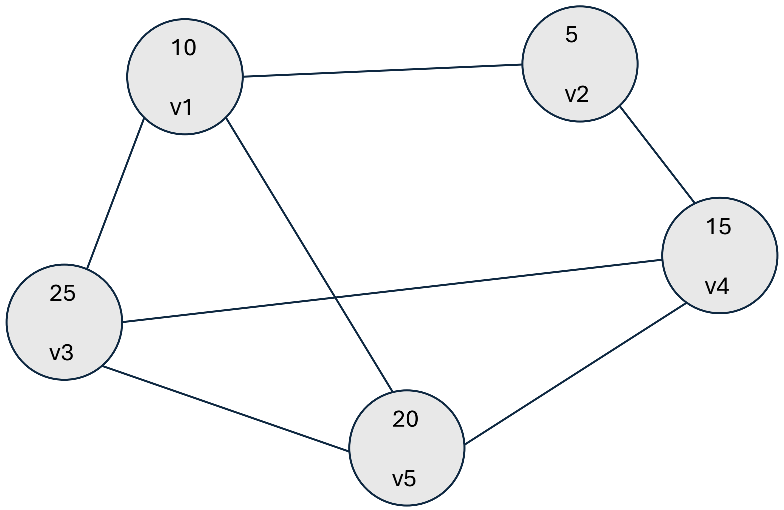
Given an undirected graph G = (V, E), a set of vertices $S \subseteq V(G)$ is a vertex cover if for $(u, v) \in E(G)$, $u \in S$ or $v \in S$ (or both).

In a weighted graph (weights on vertices) G = (V, E, Wt), a vertex cover S is called the minimum weighted vertex cover of G if Wt(S) is minimum among all possible vertex covers.

{V1, V2, V4}?

{V2, V3, V5}? = 50

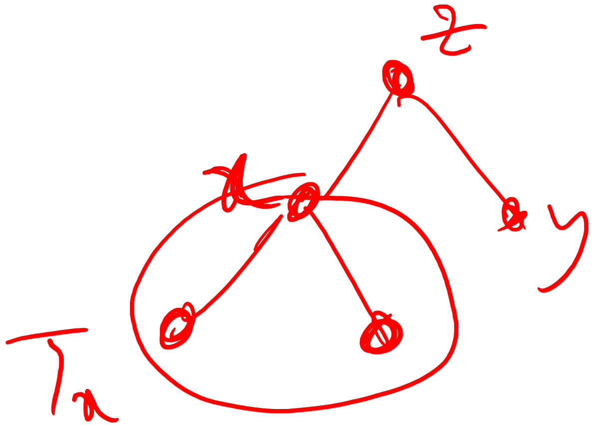
{V1, V4, V5}? = 45



Vertex Cover

Input: Weighted rooted tree. T = (V, E, Wt)
Output: Find vertex cover with minimum total weight.

Assume its a binary tree. For every vertex $x \in V(T)$, T_x be a subtree rooted at x.



Vertex Cover

Input: Weighted rooted tree. T = (V, E, Wt)
Output: Find vertex cover with minimum total weight.

Assume its a binary tree. For every vertex $x \in V(T)$, T_x be a subtree rooted at x.
For every $x \in V(T)$ define following subproblems

- MVC(x,1): Weight of a minimum weight vertex cover of T_x that contains x.
- MVC(x,0): Weight of a minimum weight vertex cover of T_x that does not contains x.
- VC(x): Minimum weight vertex cover of T_x.

Define VC(x): min (MVC(x, 1), MVC(x, 0))

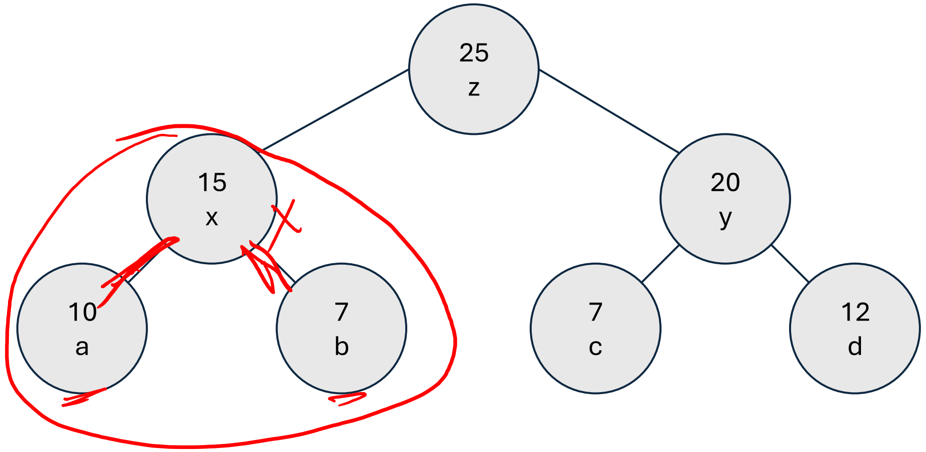
Subproblem of VC

$MVC(x, 0) = 17$

$MVC(x, 1) = 15$, $VC(x) = 15$

$MVC(y, 1) = 20$ $\quad VC(y) = 19$

$MVC(y, 0) = 19$

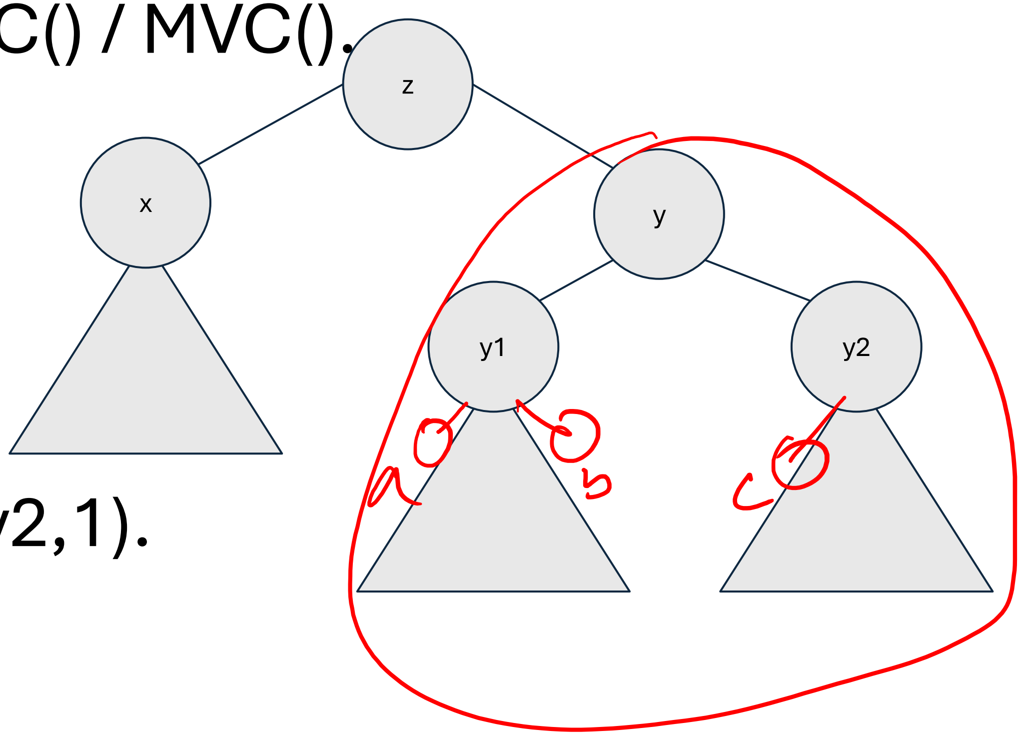


VC(x)?

Recurrence of Subproblem

How to represent MVC(y) using its child's VC() / MVC().

MVC(y,0) =



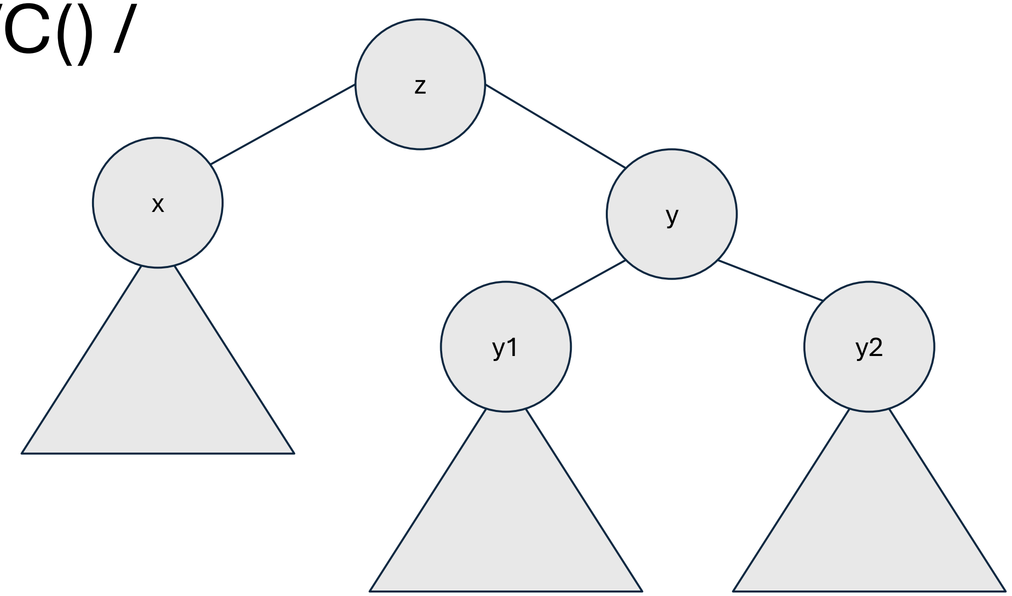
MVC(y,0) must account MVC(y1,1) & MVC(y2,1).

Recurrence of Subproblem

How to represent MVC(y) using its child's VC() / MVC().

$$\mathrm{MVC}(y,0) = \mathrm{MVC}(y1,1) + \mathrm{MVC}(y2,1)$$

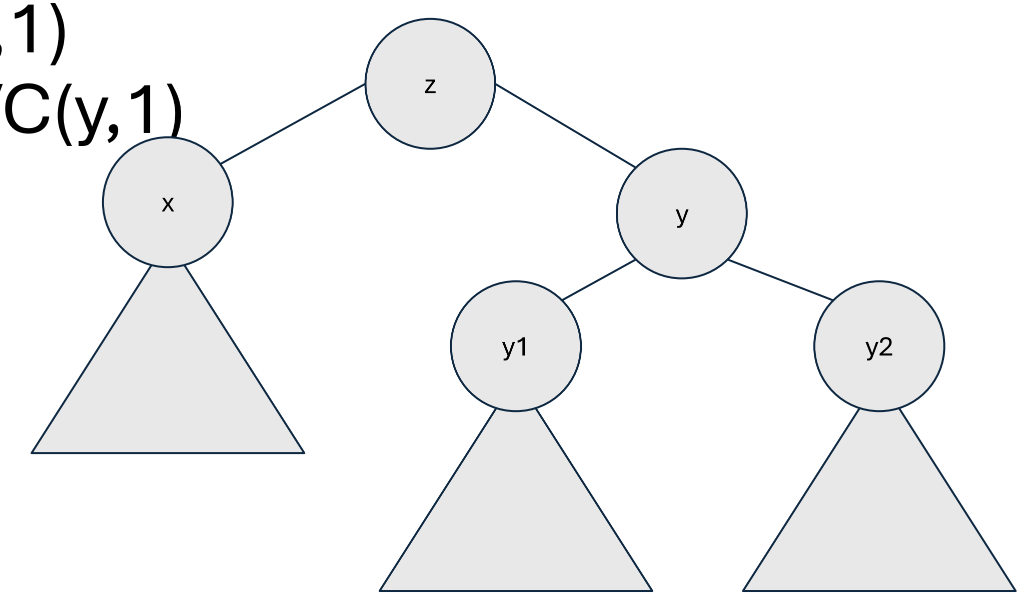
$$\mathrm{MVC}(y,1) = Wt(y) + \min \begin{cases} \mathrm{MVC}(y1,0) + \mathrm{MVC}(y2,0), \\ \mathrm{MVC}(y1,0) + \mathrm{MVC}(y2,1), \\ \mathrm{MVC}(y1,1) + \mathrm{MVC}(y2,0) \end{cases}$$



If y is a leaf then, MVC(y,0) = 0 & MVC(y,1) = Wt(y).

Algorithm

- For VC(z), compute MVC(z,0) & MVC(z,1)
- For MVC(z,0), compute MVC(x,1) & MVC(y,1)
- So on…

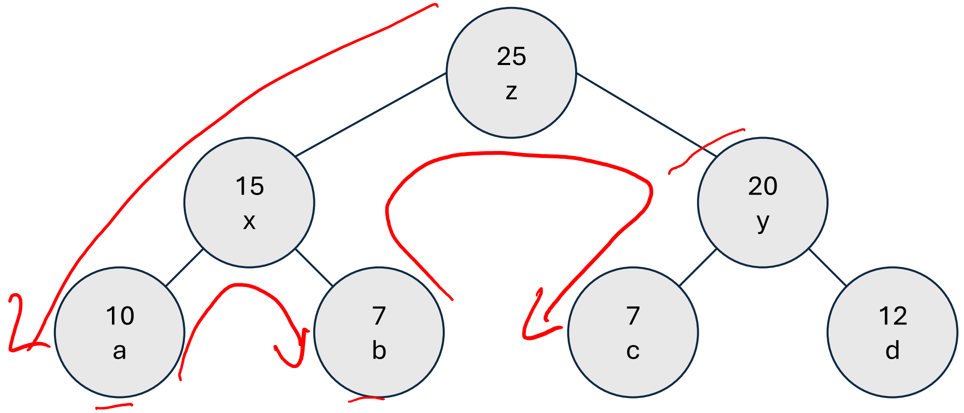


Use postorder traversal.

Vertex Cover

$$\min \begin{Bmatrix} MVC(x,0) + MVC(y,0) \\ MVC(x,1) + MVC(y,0) \\ MVC(X,0) + MVC(y,1) \end{Bmatrix} + w(z)$$

	a	b	x	c	d	y	z
0	0	0	17	0	0	19	35
1	10	7	15	2	7	20	59



Reference

Slides