November 17, 2023 at 4:55 PM

# CMPT 360 - ASN3 - 1

1

- Heaviest-first algorithm

  - A counter example is:

Given path:

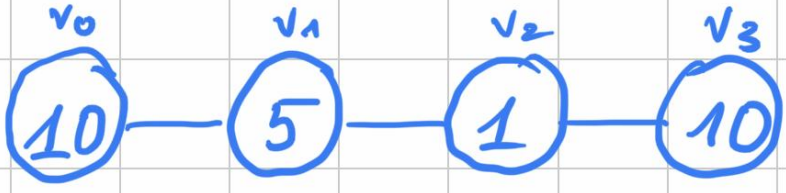

Expected maximum weight independent set: $v_0, v_2$ (total weigh = 20)

Actual returned from heaviest-first algo: $v_1$ (total weigh = 15)

- Largest independent sets

  - A counter example is:

Given path:



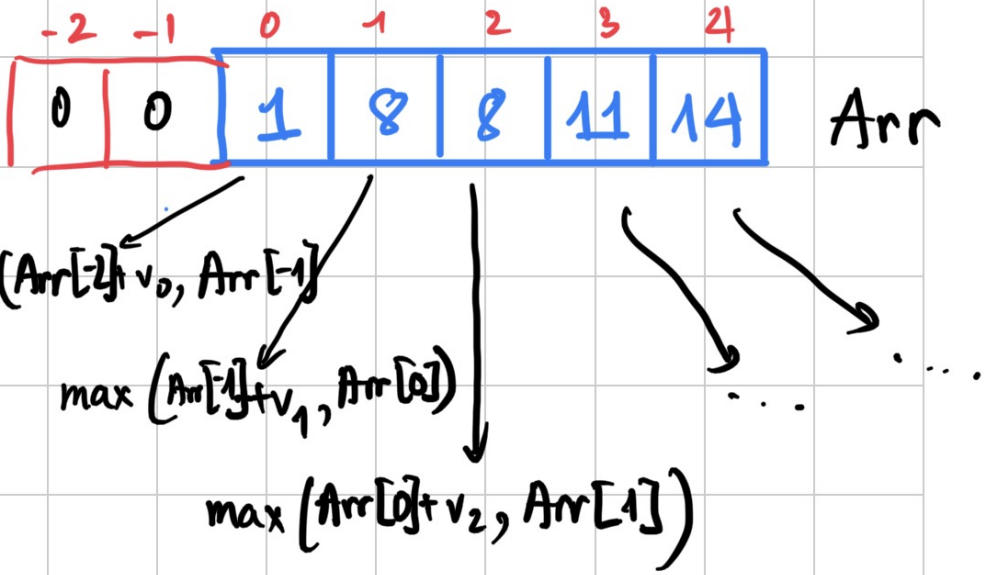
Expected set : $v_0, v_3$ (total weigh = 20)

Actual set returned by algo: $v_1, v_3$ (total weigh = 15)

- The table content of my algorithm

Given


Memoization



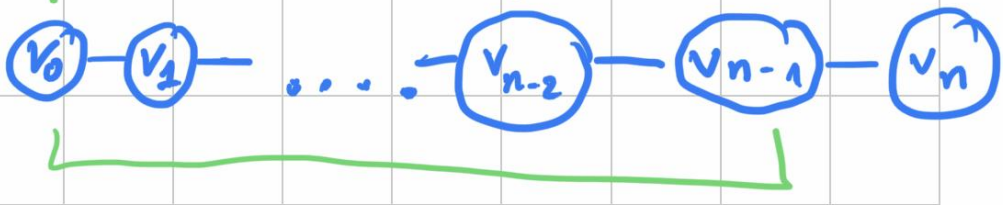

- Find an O(n) algorithm that does work

## a> ALGORITHM

Let $T(n)$ be the function that find the maximum weigh independent set of the first $n+1$ vertices of the given path $P$

↳ up to $v_n$

Given path :




Recursion idea :

$T(-2) = \emptyset \Rightarrow \text{weigh}(T(-2)) = 0$

$T(-1) = \emptyset \Rightarrow \text{weigh}(T(-1)) = 0$

& with $n \geq 1$

we know for sure that we could add $v_n$ into this set

If $\text{weigh}(T(n-1)) \geq \text{weigh}(T(n-2)) + v_n$

$T(n) = T(n-1)$

else:

$T(n) = T(n-2) \cup v_n$

## Algorithm Pseudocode:

```
# Given G = (V, E)
weigh_table := IntArray of Size (n+2)
weigh_table[-2] = 0
T_{n-2} = T_{-2}   (= ∅)
weigh_table[-1] = 0
T_{n-1} = T_{-1}   (= ∅)
for (k = 0; k < n+1; k++):
  if weigh_table[k-1] > weigh_table[k-2] + v_k:
      T_k = T_{n-1}
      weigh_table[k] = weigh_table[k-1]
  else:
      T_k = T_{n-2} ∪ v_k
      weigh_table[k] = weigh_table[k-2] + v_k
  # prepare T_{n-2} & T_{n-1} for next iteration
  T_{n-2} = T_{n-1}
  T_{n-1} = T_k
```

## b) CORRECTNESS

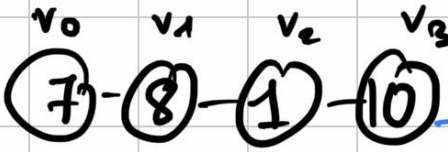
- Give a path $P = \{v_0, v_1, v_2, v_3, \dots v_n\}$
- And Maximum-weigh Independent Set: $T_n$
- Add $v_{n+1}$ to $P \Rightarrow$ Find $T_{n+1}$ ?

- **Staying Ahead**

- If including $v_{n+1}$ in $T_{n+1}$ doesn't improve the maximum weight, we don't include it

$\Rightarrow$ so that we have the option to include $v_{n+2}$ (if any)

e.g




$T_3 = v_1, v_3$

weigh$(T_1) = 8$
weigh$(T_0) + v_2 = 8$
$\Rightarrow$ including $v_2$ doesn't improve the max weigh
$\Rightarrow T_2 = v_1$

- Only including $v_{n+1}$ if it improves the maximum weigh

- **Exchange Argument**

- If algo returns $T_{n+1} = T_n$ :

## • Exchange Argument

- If algo returns $T_{n+1} = T_n$ :

(not including $v_{n+1}$ in the set)

but the optimal ($T_{n+1}^{*}$) includes $v_{n+1}$ :

($T_{n+1} = T_{n-1} \cup v_{n+1}$)

$\Rightarrow$

* If weigh($T_{n+1}$) = weigh($T_{n+1}^{*}$)
  $\Rightarrow$ $T_{n+1}$ is as good as ($T_{n+1}^{*}$)
* If weigh($T_{n+1}$) < weigh($T_{n+1}^{*}$)
  $\Rightarrow$ weigh($T_n$) < weigh($T_{n-1}$) + $v_{n+1}$
  $\Rightarrow$ The algo would have returned $T_{n+1} = T_{n-1} \cup v_{n+1}$ instead of $T_{n+1} = T_n$

## c> Prove its O(n)

* For each vertex in the path we need $T_{n-2}$, $T_{n-1}$, weigh($T_{n-2}$) and weigh($T_{n-1}$) which all have O(1) as we can just get them from the memoization table

$\Rightarrow$ O(n)