11 Oct 2021

Why is the greedy algo for trick-or-treating optimal?

- $Y_{Greedy} = [y_1^G, y_2^G, \ldots, y_n^G]$

this is the sol'n from our greedy algorithm

- $Y_{OPT} = [y_1^O, y_2^O, \ldots, y_m^O]$

this is <u>an</u> $\underbrace{\text{optimal}}$ $\underbrace{\text{sol'n.}}$

of all sol'ns, smallest # of houses to stop at $\Rightarrow m \leq n$

y_m^O = home
no 2 adjacent houses are ≥ 30 m

We want to show $m = n$. Already know $m \leq n$.

<u>Claim:</u> $\forall i$, $y_i^G \geq y_i^O$

<u>Proof:</u> Base case $i=1$. By choice of y_1^G, we selected the farthest possible $\Rightarrow y_1^G \geq y_1^O$.

Let $i \geq 1$. Assume $y_i^G \geq y_i^O$ and $i < n$, $i < m$.

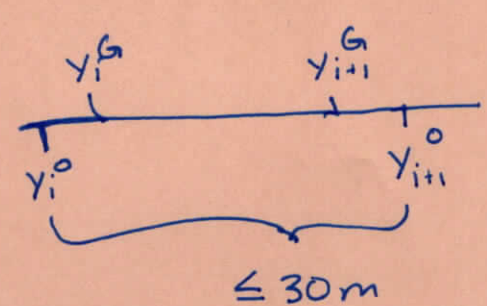
Assume, by contradiction that $y_{i+1}^O > y_{i+1}^G$.

Since y^{OPT} is a sol'n, we know $y_{i+1}^O - y_i^O \leq 30$.

If $y_i^G > y_{i+1}^O$, this is a contradiction to $y_{i+1}^O > y_{i+1}^G$

$\therefore$ we know $y_i^O < y_i^G < y_{i+1}^O$

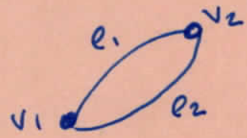
Skratch work



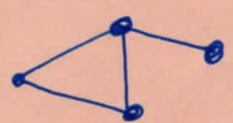
$\Rightarrow y_{i+1}^O - y_i^G \leq 30 \Rightarrow y_{i+1}^O \leq y_{i+1}^G$ ※ so, $y_{i+1}^G \geq y_{i+1}^O$, awtbs.

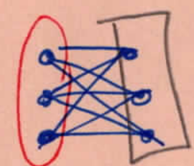
Graphs

a graph $G = (V, E)$ is two sets, V is a set of vertices and E is a set of pairs of vertices that we call edges or arcs



} is a multigraph: allows multiple edges + self-loops.

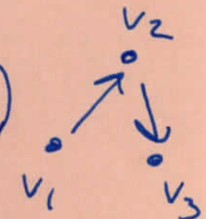




$K_{3,\boxed{3}}$: the bipartite graph on sets of size 3 and $\boxed{3}$

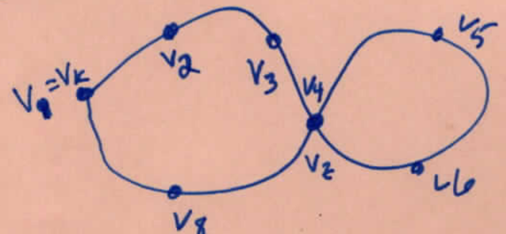
- a graph is directed if we make our edges ordered pairs of vertices instead of just pairs

ex: $G = (\{v_1, v_2, v_3\}, \{[v_1, v_2], [v_2, v_3]\})$



- a path is a sequence of vertices $[p_1, p_2, \ldots, p_k]$ such that $\{p_i, p_{i+1}\} \in E$ for all $i = 1, \ldots, k-1$.

- a cycle is a path that starts & ends at the same vertex (aka a "closed path")



$v_7 = v_4$, so not a cycle according to JE but is a cycle according to BTF

- a graph is acyclic iff it does not contain any simple cycles (those w/out repeated ~~vertices~~ edges)

Examples of where we use graphs to solve problems:

- road networks
- resource delivery logistics (directed, weighted graph)
- databases / ER diagrams
  every set of ~~in~~ data is its own node
- circuits
- ecology
- social networks
- flow networks
  - → fish in streams
  - → network traffic
  - → actual traffic
- neural networks
- neighborhood graphs
  - → nodes: data & we know the complete weighted graph, where weights = distances btwn data pts.
  - → nbhd graph: includes all edges $\leq t$, where t is some threshold

  e.g., data = circles in the plane

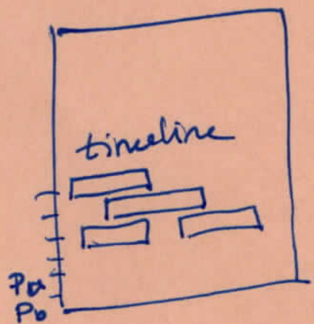
  

  } node for each circle
  } edge if 2 circles intersect

problems/algorithms

- Eulerian tour: a path that ~~does not use an~~ ~~ed~~ uses each edge exactly once
  (e.g., bridges of Königsberg)
- Hamiltonian path: a path that uses each vertex exactly once.
- vertex cover
- 4-colorability / k-colorability
- traveling salesman (NPC)
- minimum spanning tree
  (= a tree that contains all nodes of the graph)

example at a start-up



Q: how do I best color-code the timeline so adjacent rows are not the same color?

transformed this Q into a graph coloring problem!

The four color theorem

Given a plane graph, the vertices can be colored by 4 colors where no 2 adjacent vertices are the same color.

More generally

- A graph is k-colorable iff $\exists$ an assignmt of vertices to k colors such that no 2 adjacent vertices are the same color.