Algorithms

Kiho Choi

Fall, 2022

Department of AI·Software
Gachon University

4. Greedy Algorithms II

Contents

Part 2

- Counting money
- Minimum spanning tree
- Traveling salesman
- Other greedy algorithms

Counting money

- Suppose you want to count out a certain amount of money, using the fewest possible bills and coins
- A greedy algorithm would do this would be:
  At each step, take the largest possible bill or coin that does not overshoot
  - Example: To make $6.39, you can choose:
    - a $5 bill
    - a $1 bill, to make $6
    - a 25¢ coin, to make $6.25
    - A 10¢ coin, to make $6.35
    - four 1¢ coins, to make $6.39

0.01 $= 1 cent = 1 penny

0.05 $= 5 cent = 1 nickel

0.01 $= 10 cent = 1 dime

0.01 $= 25 cent = 1 quarter

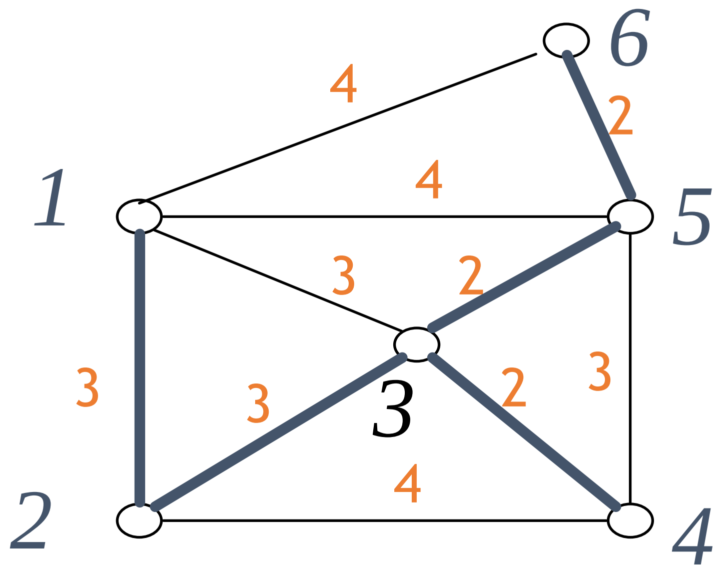
- For US money, the greedy algorithm always gives the optimum solution

A failure of the greedy algorithm

- In some (fictional) monetary system, “krons” come in 1 kron, 7 kron, and 10 kron coins
- Using a greedy algorithm to count out 15 krons, you would get
  - A 10 kron piece
  - Five 1 kron pieces, for a total of 15 krons
  - This requires six coins
- A better solution would be to use two 7 kron pieces and one 1 kron piece
  - This only requires three coins
- The greedy algorithm results in a solution, but not in an optimal solution

Minimum spanning tree

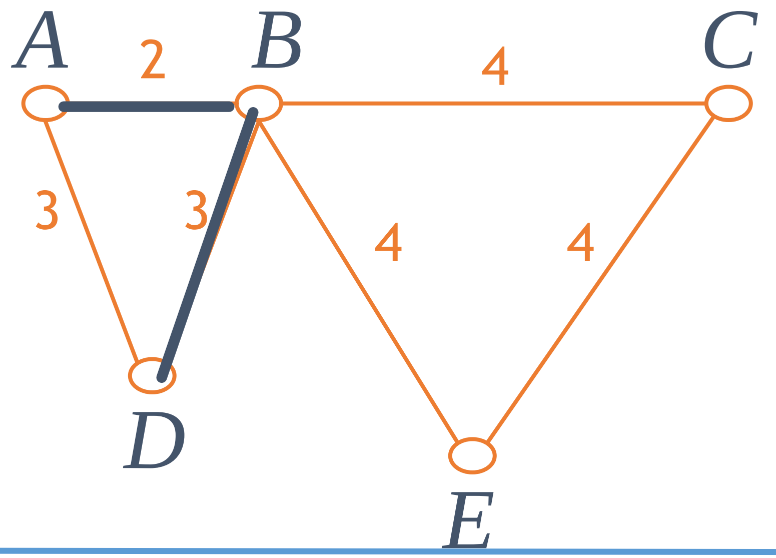
- A minimum spanning tree is a least-cost subset of the edges of a graph that connects all the nodes
  - Start by picking any node and adding it to the tree
  - Repeatedly: Pick any *least-cost* edge from a node in the tree to a node not in the tree, and add the edge and new node to the tree
  - Stop when all nodes have been added to the tree



- The result is a least-cost (3+3+2+2+2=12) spanning tree
- If you think some other edge should be in the spanning tree:
  - Try adding that edge
  - Note that the edge is part of a cycle
  - To break the cycle, you must remove the edge with the greatest cost
    - This will be the edge you just added

Traveling salesman

- A salesman must visit every city (starting from city ***A***), and wants to cover the least possible distance
  - He can revisit a city (and reuse a road) if necessary
- He does this by using a greedy algorithm: He goes to the next nearest city from wherever he is

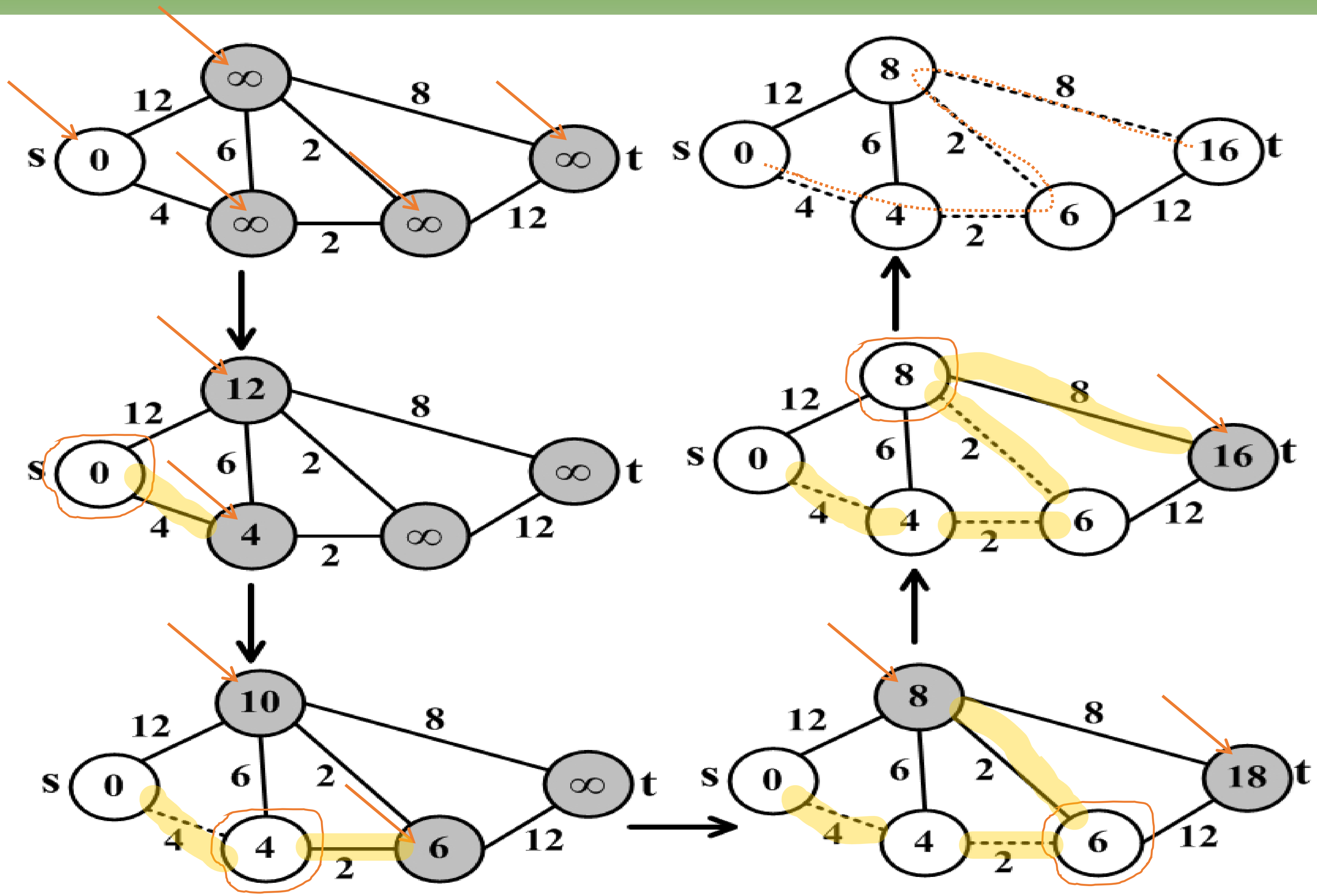


- From ***A*** he goes to ***B***
- From ***B*** he goes to ***D***
- This is *not* going to result in a shortest path!
- The best result he can get now will be ***ABDBCE***, at a cost of 16
- An actual least-cost path from ***A*** is ***ADBCE***, at a cost of 14

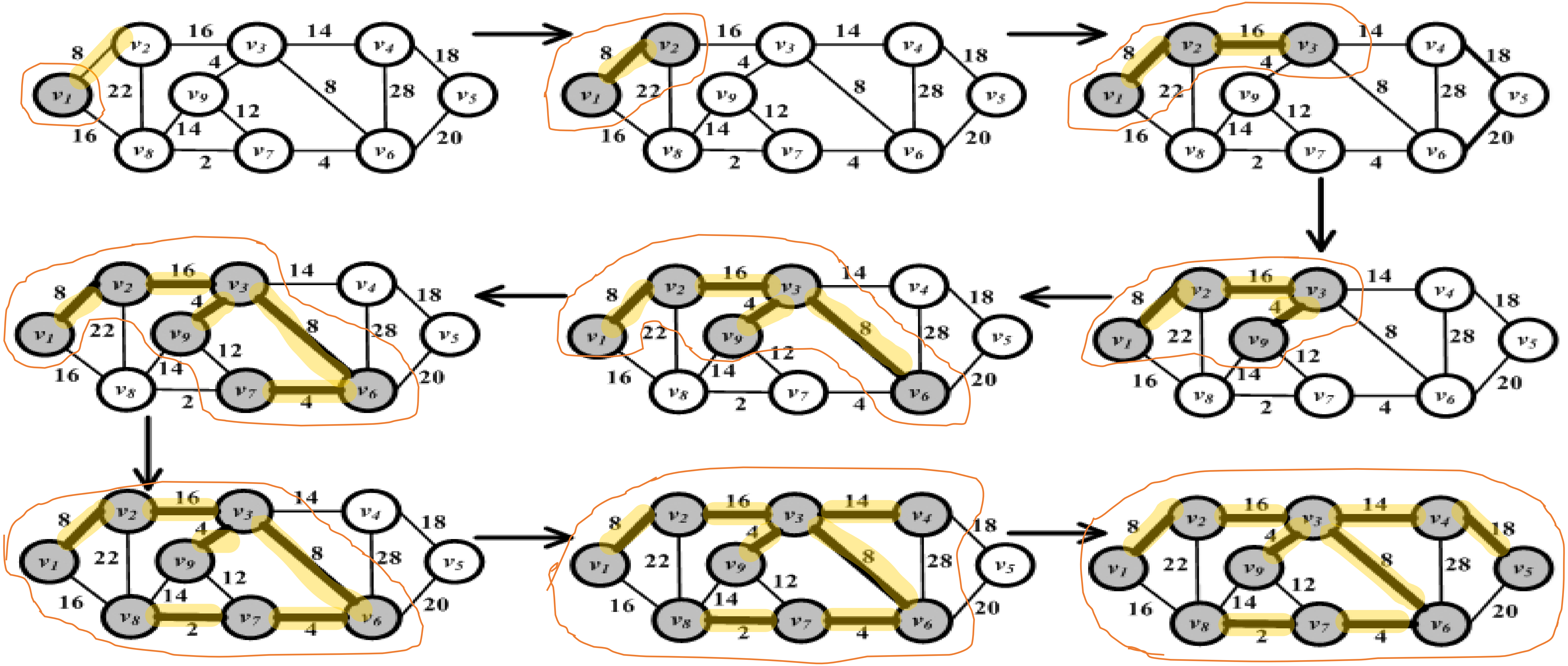
Other greedy algorithms

- Dijkstra’s algorithm for finding the shortest path in a graph
  - Always takes the *shortest* edge connecting a known node to an unknown node
- Prim’s algorithm for finding a minimum-cost spanning tree
  - Always takes the *lowest-cost* edge between nodes in the spanning tree and nodes not yet in the spanning tree

Dijkstra's Algorithm Execution Example



Prim’s Algorithm Execution Example



THANK YOU