CAD Algorithms

Greedy Algorithms

Mohammad Tehranipoor

ECE Department

Greedy Algorithms

- Greedy algorithms work in phases.
- In each phase, a decision is made that appears to be good, without regard for future consequences.
- Generally, this means that some **local optimum** is chosen.
- This '**take what you can get now**' strategy is the source of the name for this class of algorithms.
- Greedy algorithms always take the best immediate solution.

Greedy Algorithms

- When the algorithm terminates, we **hope** that the local optimum is equal to the **global optimum**.
- If this is the case, then the algorithm is correct; otherwise, the algorithm has produced a suboptimal solution.

Recommendation

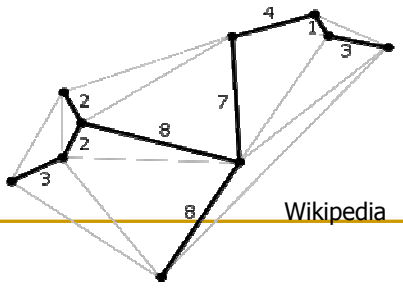
- If the best answer is not required, then simple greedy algorithms are sometimes used to generate approximate answers, rather than using the more complicated algorithms generally required to generate an exact answer.

Cont.

- Greedy algorithms are usually **quicker**, since they don't consider the details of possible alternatives.
- Greedy algorithms are **simple** and **straightforward**.
- They are straightforward in the sense that they make decisions without being worried about the impact of the decision on future decisions.
- Greedy algorithms never **reconsider** the decisions made so far.

Cont.

- Greedy algorithms may find some optimal solutions for some optimization problems, but may find less-than-optimal solutions for some instances of other problems.
- Example:
  - Prim's algorithm and Kruskal's algorithm used for **Minimum Spanning Tree (MST)** are greedy algorithms that find globally optimal solution, an MST.
  - Dijkstra's algorithm for finding **Shortest Path** is another greedy algorithm.



Wikipedia

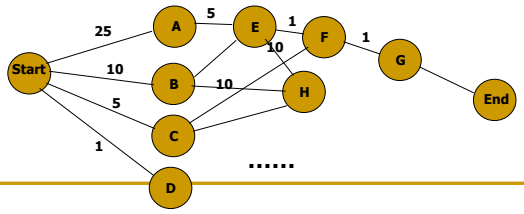
Structure of Greedy Algorithm

- Initially the set of chosen items is empty i.e., solution set.
- At each step
  - Item will be added in a solution set by using **selection function**.
  - IF the set would no longer be feasible
    - reject item under consideration (and is never considered again).
  - ELSE IF set is still feasible THEN
    - add the current item.

A Simple Example

- Cashier: Making Change
  - A cashier does not consider all the possible ways in which to count a given sum of money. Instead, he/she starts with largest denomination and proceeds to the smallest.

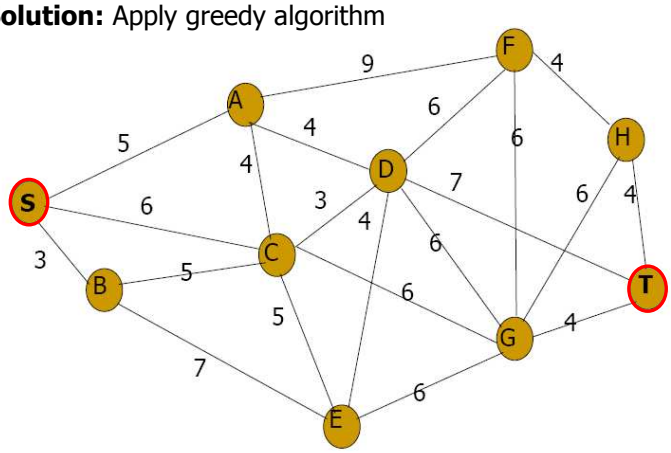
- Example: 32 cents
  - Start with quarter, then add a nickel and then add two pennies.
  - Of course he/she makes decisions based on the availability of coins.



Shortest Path Problem

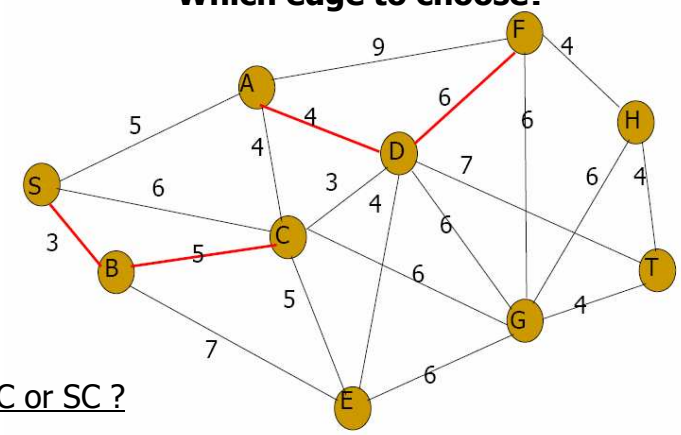
Problem: Find the shortest path between S and T

Solution: Apply greedy algorithm



Shortest Path

Which edge to choose?

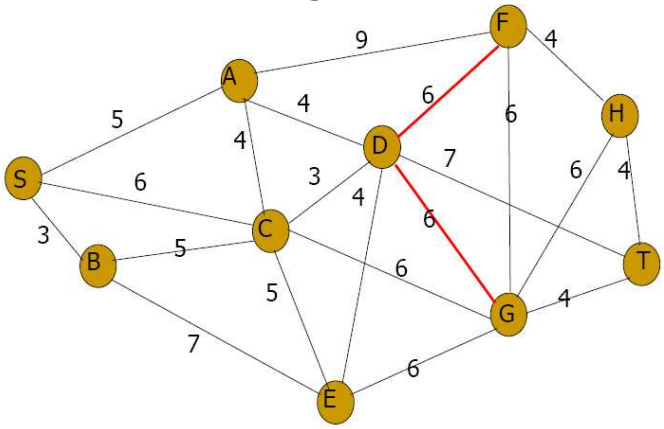


SBC or SC ?

ADF or AF ?

Shortest Path

Which edge to choose?



DF or DG ?
Both have same weight.

Heuristic Algorithms

Heuristic Algorithms

- The term **heuristic** is used for algorithms which find solutions among all possible ones, but they **do not guarantee** that the best will be found.
- Heuristics may be considered as approximately and not accurate algorithms.
- These algorithms, usually find a solution close to the best one and they find it fast and easily.
- Sometimes these algorithms can be accurate, that is they actually find the best solution, but the algorithm is still called heuristic until this best solution is **proven** to be the best.

Heuristic Algorithms

- In some cases, we may not be interested in the optimal solution:
  - Because the size of problem that we want to solve is beyond the computational limit of known optimal algorithms within the computer time we have available.
  - We could solve optimally but feel that this is not worth the effort (time, money, etc) we would spend in finding the optimal solution.
- In such cases we can use a heuristic algorithm.
- A **well-designed heuristic** algorithm can give good quality (near-optimal) results.

Example

- Consider the following problem: 3 men are to be assigned to 3 jobs - where the assignment cost is given by the matrix below:

M \ J	1	2	3
A	1	3	4
B	3	7	4
C	3	4	2

- Only one man can be assigned to one job and all the men should be assigned. What would be a heuristic algorithm for this problem?

Cont.

- One (simple) heuristic for the assignment problem would be:
  - Choose a man and a job at random.
  - Assign the chosen man to the chosen job.
  - Delete the chosen man and the chosen job from the problem and **repeat** with this new (smaller) problem.
- This heuristic does not use any of the cost information and so we would not expect it to give very good results.

Man B Job 2 Cost 7
Man C Job 1 Cost 3
Man A Job 3 Cost 4
Total Cost: 14

M \ J	1	2	3
A	1	3	4
B	3	7	4
C	3	4	2

Cont.

- A better heuristic might be:
  - Choose the smallest cost in the cost matrix (ties broken arbitrarily) and assign the corresponding man to the corresponding job.
  - Delete them from the problem and repeat with this new (smaller) problem.
- This heuristic, actually, is a greedy algorithm.

Cost 1 Assign A to job 1
Cost 2 Assign C to job 3
Cost 7 Assign B to job 2
Total Cost: 10

M \ J	1	2	3
A	1	3	4
B	3	7	4
C	3	4	2

Optimal Cost : 8 , A: 3, B: 3, C: 2