BLG 336E – Analysis of Algorithms II

Practice Session 2

Atakan Aral

Istanbul Technical University – Department of Computer Engineering

18.03.2014

Outline

Greedy Algorithm

An algorithm that builds a solution in small steps, choosing a decision at each step myopically [=locally, not considering what may happen ahead] to optimize some underlying criterion.

- Produces optimum solution for some problems.
  - Minimum spanning tree
  - Single-source shortest paths
  - Huffman trees
- Produces good aproximate solutions for some other problems.
  - NP-Complete problems such as graph coloring

- You own a coffe shop that has n customers.
- It takes t_i minutes to prepare coffee for the i^{th} customer.
- i^{th} customer's value for you (i.e. how frequent s/he comes to your shop) is v_i.
- If you start preparing coffee for the i^{th} customer at time s_i, you finish at $f_i = s_i + t_i$.
- All customers arrive at the same time.
- You can prepare one coffee at a time.
- There is no gap after you finish one coffee and start another.

- You need to design an algorithm.
- **Input:** n, t_i, v_i
- **Output:** A schedule (i.e. ordering of customer reqests)
- **Aim:** Minimize wait time especially for valued customers

$$Minimize : \sum_{i=1}^{n} f_i * v_i \tag{1}$$

- What is the time complexity of your algorithm?
- Run your algorithm for a sample input.

Solution

```
input  : t[], v[], n
for i ← 1 to n do
    w[i, 1] ← v[i]/t[i];          // weight of each customer
    w[i, 2] ← i;
sort(w, dec, 1);
t ← 0;
cost ← 0;
for j ← 1 to n do
    schedule[j] ← w[j, 2];
    f[j] ← t + t[schedule[j]];
    t ← f[j];
    cost ← cost + f[schedule[j]] ∗ v[schedule[j]];
return schedule, f, cost
```

Solution

- Both for loops take $O(n)$ time.
- Complexity of the algorithm depends on sort method.
- Typically $O(nlogn)$

- Input:
  - $t_1 = 2, t_2 = 3, t_3 = 1$
  - $v_1 = 10, v_2 = 2, v_3 = 1$
- Output:
  - Weights = 5, 0.67, 1
  - Schedule: 1, 3, 2
  - Finish times: 2, 3, 6
  - Cost: 2*10 + 3*1 + 6*2 = 35

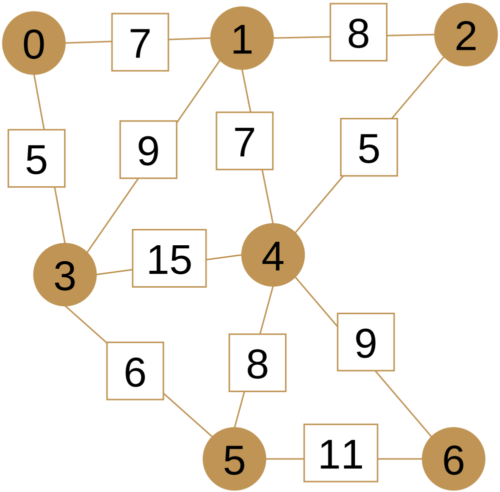
- Local goverment wants to reduce operating costs of road lighting.
- Not every road will be illuminated at night.
- For safety, there will be at least one illuminated path between all junctions.

- Suggest an algorithm to optimize the road lighting
- What is the maximum saving without endangering the citizens?

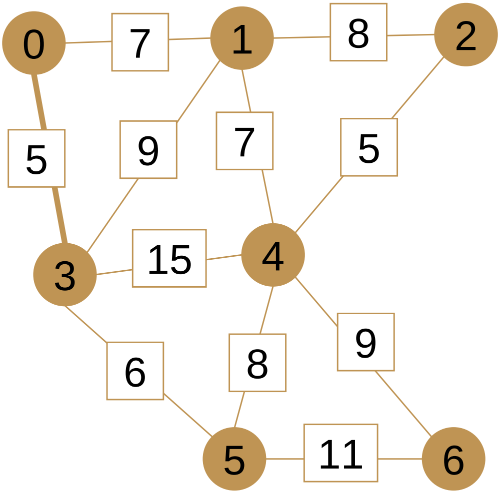
- Input contains the junctions and the roads connecting them
- As well as, the cost of illuminating each road.
- File format is as follows:

```
7 11
0 1 7
0 3 5
1 2 8
1 3 9
1 4 7
2 4 5
3 4 15
3 5 6
4 5 8
4 6 9
5 6 11
```

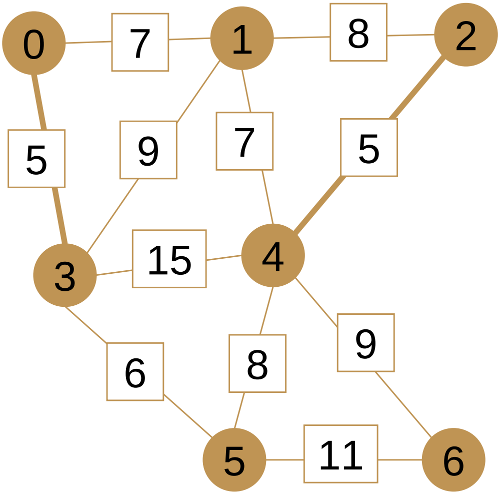
Solution



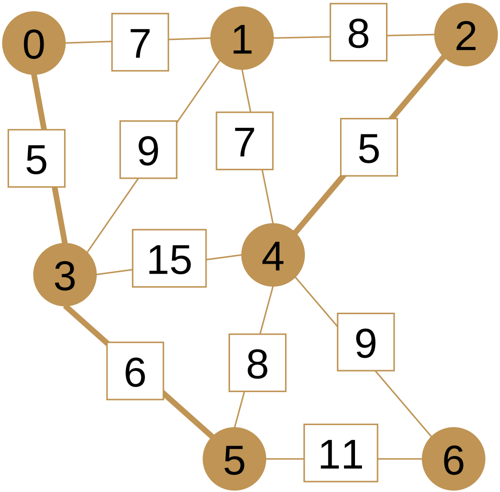
Solution



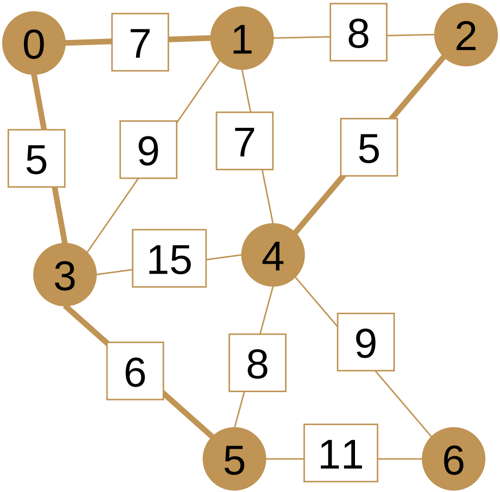
Solution



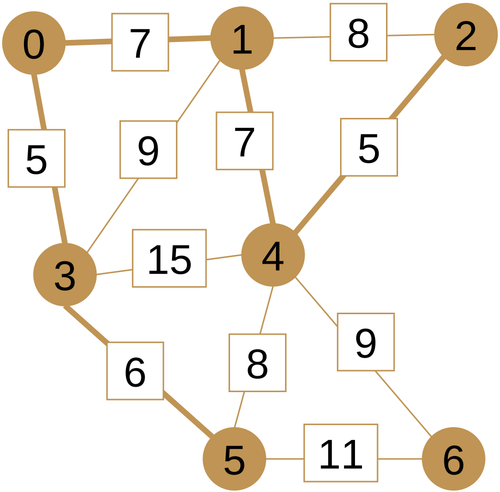
Solution



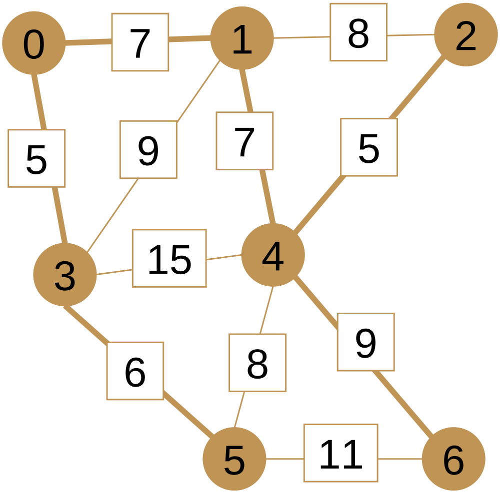
Solution



Solution



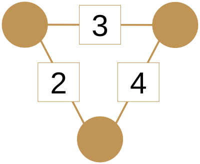
Solution



Total cost: 90 – **Optimized cost:** 39 – **Saving:** 51

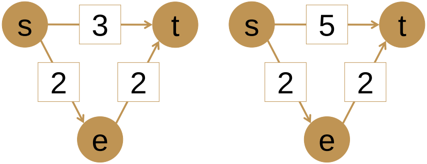
True or False?

- Let G be an arbitrary connected, undirected graph with a positive distinct cost $c(e)$ on every edge e.
- Let T be a MST of G. If we replace edge cost $c(e)$ with $c(e) * c(e)$, T must still be MST for G.
- Same is valid for $c(e) + 5$.
- It is also valid if negative costs are allowed.



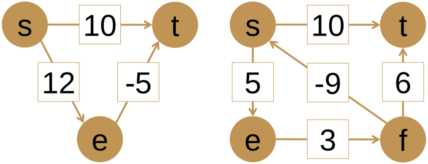
- MST depends only on the order of the costs, actual values are not important as long as order is the same.

- Let G be an arbitrary connected, directed graph with a positive cost $c(e)$ on every edge e.
- Let P be the shortest path between node s and node t in G. If we replace edge cost $c(e)$ with $c(e)^2$, P must still be shortest path between node s and node t in G.
- Same is valid for $c(e) + 5$.



- For shortest paths, actual values of the costs do matter.

- Let G be an arbitrary connected, directed graph with cost $c(e)$ on every edge e.
- Dijkstra algorithm finds the correct solution if some edges have negative costs.
- There always exists a solution even if Dijkstra algorithm cannot find it.



- There is no solution for graphs with negative cycles.

$s \rightarrow BLG101E \rightarrow BLG102E \rightarrow BLG201E$
$BLG101E \rightarrow BLG301E \rightarrow BLG305E$
$BLG102E \rightarrow BLG303E$
$BLG201E \rightarrow BLG301E$
$BLG202E \rightarrow BLG301E$
$BLG301E \rightarrow BLG305E$
$BLG303E$
$BLG305E \rightarrow BLG202E$

1. Draw the graph G, showing its nodes and directed edges.
2. Is this graph a DAG (directed acyclic graph)? Give the reason for your answer.
3. If G is not a DAG, how can you transform it into a DAG? Which operation is required?
4. Using the DAG course graph, produce a course program for a student with the minimum number of semesters.