BLG 336E – Analysis of Algorithms II

Practice Session 3

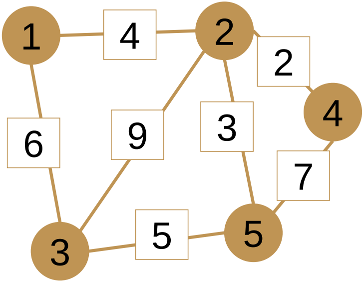
Atakan Aral

Istanbul Technical University – Department of Computer Engineering

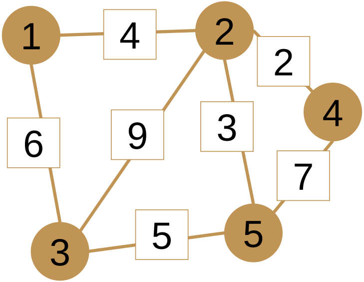
25.03.2014

Outline

- **[Midterm 2013]** Compute the shortest path from node 4 to node 1 in the following undirected graph using the greedy algorithm we learned in the class.



Solution



- $d(4) : \{\infty, \infty, \infty, 0, \infty\}, s : \{4\}$
- $d(4) : \{\infty, \mathbf{2}, \infty, 0, 7\} s : \{4, 2\}$
- $d(4) : \{6, 2, 11, 0, \mathbf{5}\}, s : \{4, 2, 5\}$
- $d(4) : \{\mathbf{6}, 2, 10, 0, 5\}, s : \{4, 2, 5, 1\}$
- $d(4) : \{6, 2, \mathbf{10}, 0, 5\}, s : \{4, 2, 5, 1, 3\}$

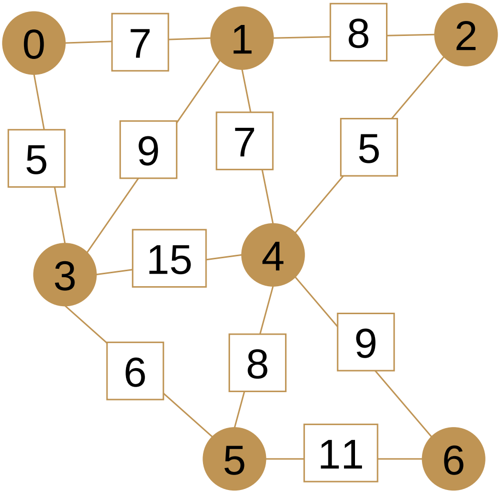
- Local government wants to reduce operating costs of road lighting.
- Not every road will be illuminated at night.
- For safety, there will be at least one illuminated path between all junctions.

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

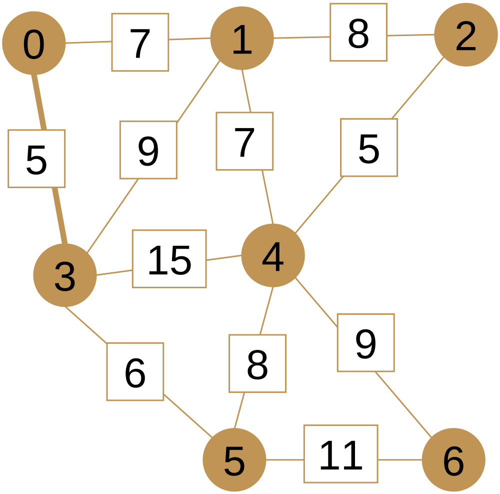
- Input contains the junctions and the roads connecting them
- As well as, the cost of illuminating each road.
- File format is as follows:

```
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
```

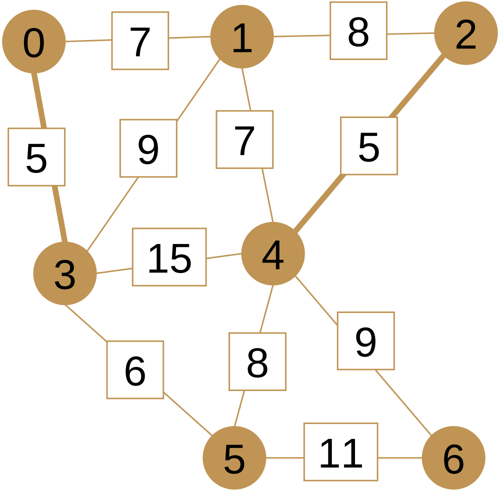
Solution



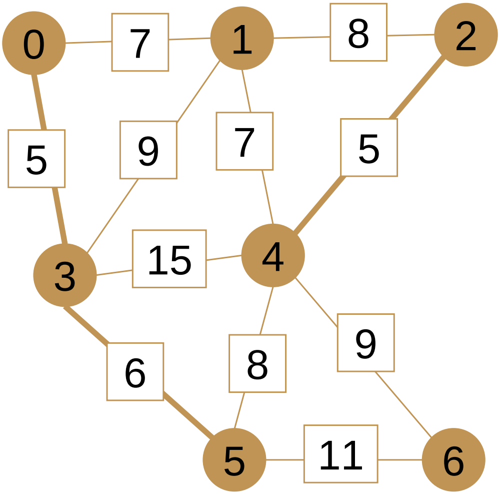
Solution



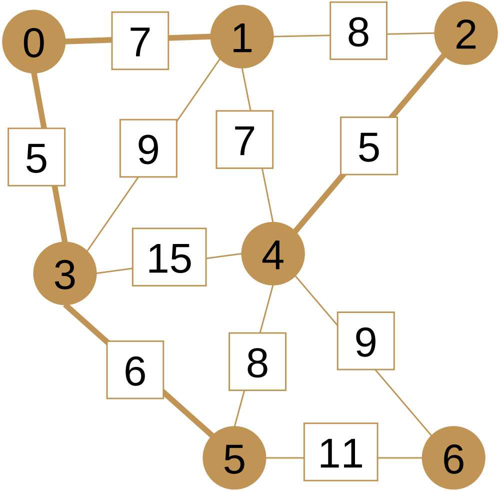
Solution



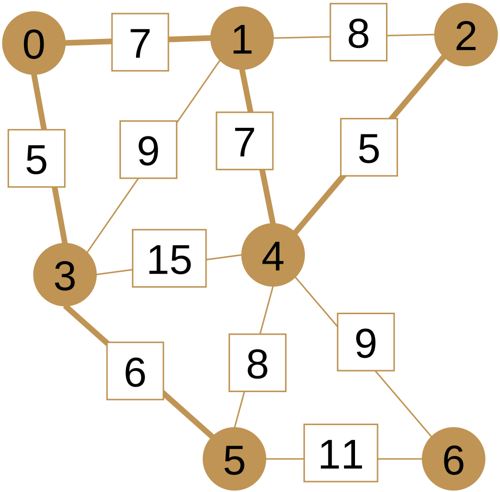
Solution

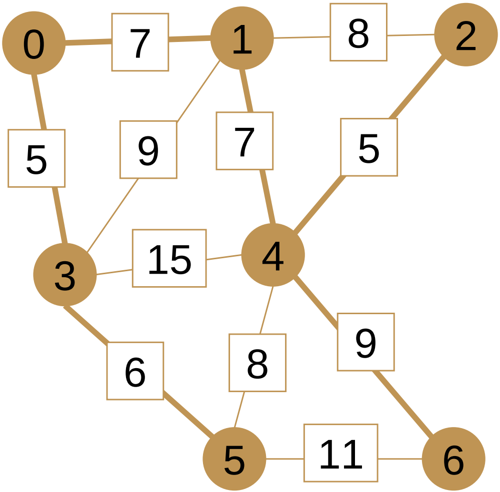


Solution



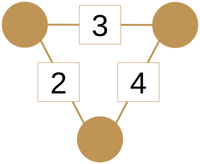
Solution





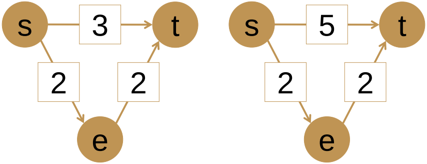
Total cost: 90 – **Optimized cost:** 39 – **Saving:** 51

- 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.

- You are interested in analyzing some hard-to-obtain data from two separate databases *A* and *B*.
- Each database contains n numerical values and you may assume that no two values are the same.
- Determine the median of this set of 2n values

- Only way you can access these values is through queries to the databases: In a single query, you can specify a value *i* to one of the two databases, and the chosen database will return the i^{th} smallest value that it contains.
- Since queries are expensive, you would like to compute the median using as few queries as possible.
- Give an algorithm that finds the median value using at most $O(logn)$ queries.

Formulation

Let $k = \lceil n/2 \rceil$ then $A(k)$ and $B(k)$ are the medians of databases A and B. We are looking for $C(n)$ where C is the merged database.

- If $A(k) < B(k)$ then:
  - $B(k) > A(1), A(2), ..., A(k)$
  - $B(k) > B(1), B(2), ..., B(k-1)$
- $B(k)$ is greater than at least $2k - 1$ elements.
- $B(k) \geq C(n)$ so no need to consider $B(k+1), B(k+2), ..., B(n)$

Formulation

Let $k = \lceil n/2 \rceil$ then $A(k)$ and $B(k)$ are the medians of databases A and B. We are looking for $C(n)$ where C is the merged database.

- If $A(k) < B(k)$ then:
  - $A(k) < B(k), B(k+1), ..., B(n)$
  - $A(k) < A(k+1), A(k+2), ..., A(n)$
- $A(k)$ is less than at least $2n - 2k - 1$ elements.
- $A(k) \leq C(n)$ so no need to consider $A(1), A(2), ..., A(k-1)$

- Otherwise, if $A(k) > B(k)$ then:
  - $A(k) > B(1), B(2), ..., B(k)$
  - $A(k) > A(1), A(2), ..., A(k-1)$
- $A(k) \geq C(n)$ so no need to consider $A(k+1), A(k+2), ..., A(n)$
- Similarly,
  - $B(k) < A(k), A(k+1), ..., A(n)$
  - $B(k) < B(k+1), B(k+2), ..., B(n)$
- $B(k) \leq C(n)$ so no need to consider $B(1), B(2), ..., B(k-1)$

median: $n, a = 0, b = 0$

if $n == 1$ **then**
  return $min(A(1), B(1))$
end
$k = \lceil n/2 \rceil$;
if $A(a+k) < B(b+k)$ **then**
  return $median(k, a + \lfloor n/2 \rfloor, b)$
end
else
  return $median(k, a, b + \lfloor n/2 \rfloor)$
end

- Array size is halved at each recursion. So the complexity is $O(logn)$.