Design and Analysis of Algorithms

L29: Bellman-Ford algorithm

Dynamic Programming

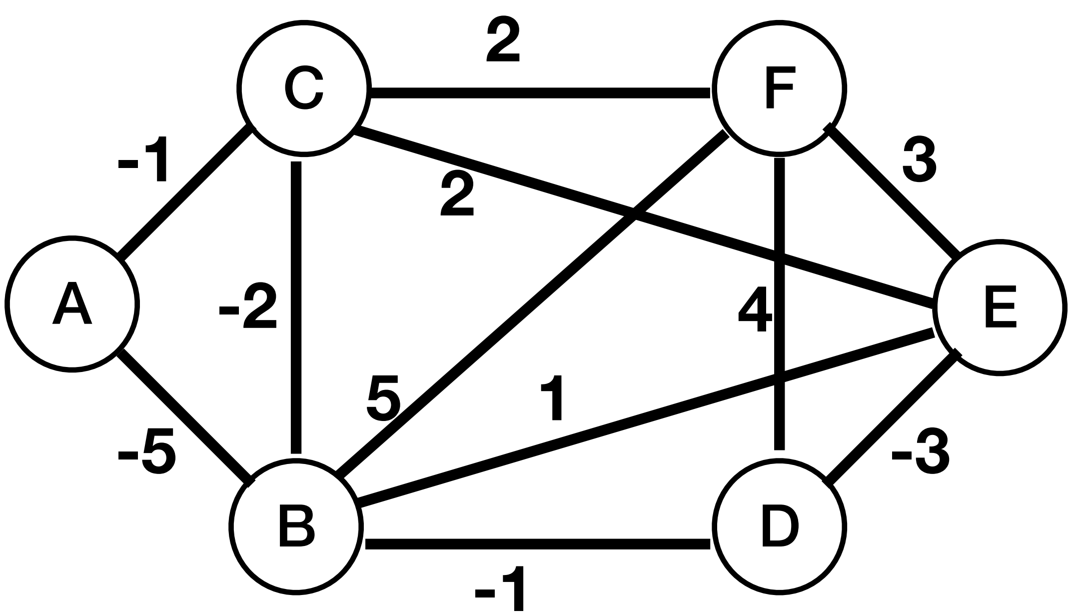
Dr. Ram P Rustagi
Sem IV (2020-Even)
Dept of CSE, KSIT
rprustagi@ksit.edu.in

Resources

- Text book 2: Horowitz
  – Sec **5 . 4**
- R1: Introduction to Algorithms
  - Cormen et al.

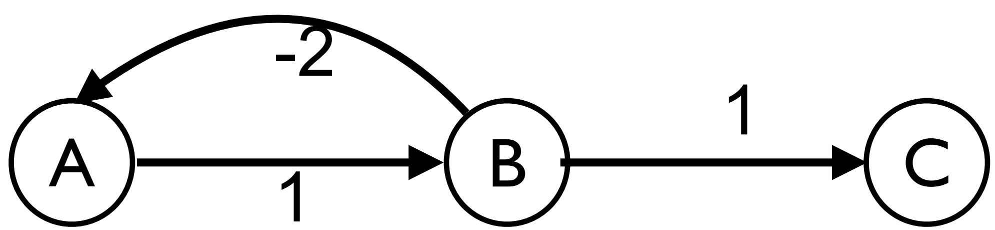
Single Source Shortest Paths

- Issues with Dijkstra's algo
  - Does not work with negative edge weights
  - Consider a graph below
    - What is Shortest distance `d(F,B)`
    - Dijkstra algo gives `0`.
    - Answer is -4



Graph with Negative edges

- Key requirement:
  - The shortest path must consist of finite number of edges.
  - Essentially, there should be no negative cycles.
  - Consider the graph below.
    - Length of shortest path from `A` to `C`
      - Is it `2` ?
    - It is $-\infty$, Length of path is `A`, `B`, `A`, `B`, ..., `A`, `B`, `C`



Single Source Shortest Path

- Consider a graph G with `n` nodes.
  - If a path has more than `n-1` edges,
    - It must contain a cycle (at least `1` vertex is repeated)
  - Elimination of path results in a shorter path
    - With same source and destination
  - When there are no negative cycles
  - Then there can be at most `n-1` edges in the shortest path from the source to the farthest node
    - This path will include all the nodes of the graph
  - Minimum path length will correspond to `1` edge
  - Use the path length as the dynamic programming approach

Single Source Shortest Path

- Consider source vertex as `s`.
- Let $\mathrm{dist}^k(v)$ denote the length of shortest path from vertex `v` from `s` containing at most `k` edges.
- Thus, $\mathrm{dist}^k(v) = \mathrm{cost}[s][v],\ 1 \le v \le n$.
- When there are no negative cycles, then
  - Restrict the search for shortest paths to length $n-1$
  - Such a length would be $\mathrm{dist}^{n-1}(v)$.

Dynamic Programming Approach

- If the shortest path from `s` to `u` with at most `k` edges, has no more than `k−1` edges, then
  - $\text{dist}^k(u) = \text{dist}^{k-1}(u)$
- If the shortest path from `s` to `u` with at most `k` edges, has exactly `k` edges, then
  - It must have shortest path from `s` to some vertex `j` followed by edge `(j,u)`.
  - The shortest path from `s` to `j` will have `k−1` edges
    - With its length as $\text{dist}^{k-1}(j)$
    - All vertices `i` such that edge `(i,v)` is in the graph will be candidate for vertex `j`.
  - Vertex `i` that minimizes ($\text{dist}^{k-1}(i)+\text{cost}[i][u]$) is the right vertex for shortest path from `s` to `u`

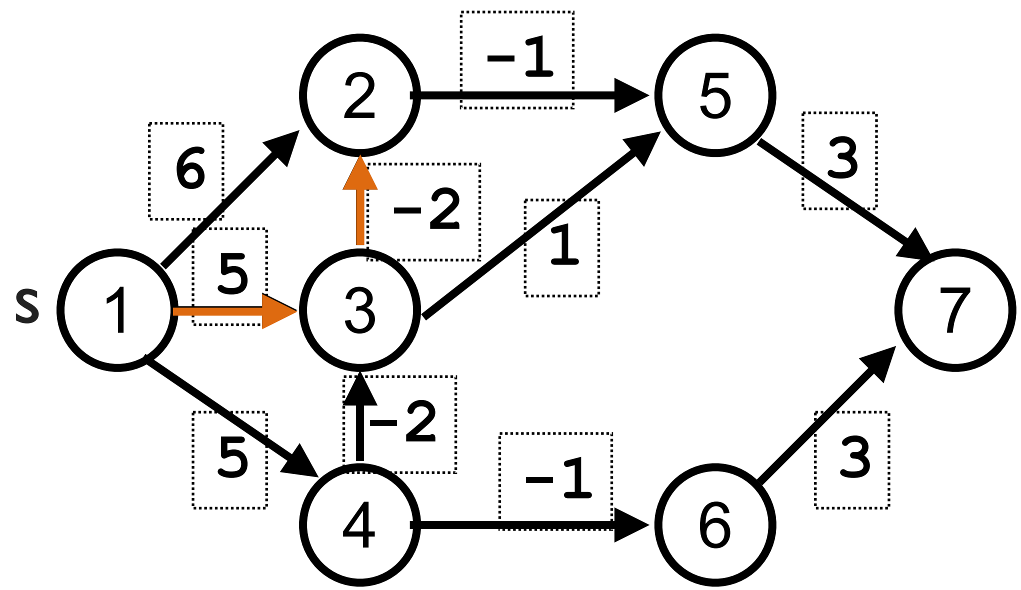
Dynamic Programming Approach

- The recurrence equation for shortest path

  $\text{dist}^{k}(u)=\min\{\text{dist}^{k-1}(u),$
  $\min_i\{\text{dist}^{k-1}(i)+\text{cost}[i][u]\}\}$
- Use the recurrence equation to compute shortest path $\text{dist}^{k}(u)$ for $k=2,3,\ldots,n-1$
- Approach involves
  - Use adjacent matrix for cost
  - First compute all shortest paths of lenth $k=2$
  - Then compute shortest paths of length $k=3$

  :
  - Finally, compute shortest paths of length $k=n-1$

Example: DP Approach



`dist^k[1..7]`

k↓	1	2	3	4	5	6	7
1	0	6	5	5	∞	∞	∞
2	0	3	3	5	5	4	∞
3	0	1	3	5	2	4	7
4	0	1	3	5	0	4	5
5	0	1	3	5	0	4	3
6	0	1	3	5	0	4	3

Algo: Bellman Ford

Algo `BellmanFord(v, n, cost[][],dist[])`
//computes single source all dstn shortest with -ve edge costs
// i/p: source `v`, number of nodes: `n`

```
for i=1 to n
   dist[i]=cost[v][i]  //default to ∞ when no edge
for k=2 to n-1
  for (each u such that u≠v, and u has incoming edge)
     for each edge <i,u> in the graph
        if dist[u] > dist[i]+cost[i][u] then
           dist[u]=dist[i]+cost[i][u]
        fi //end if
     // end for each edge
  // end for each u≠v
// end for k
```

Time Complexity: Bellman Ford

- Using adjancency matrix
  - 3 nested for loops
    - first `n-1` times (`k=2` to `n-1`)
    - other could potentially run n times.
  - Time complexity: $O(n^3)$
- Using adjancy matrix
  - outer most nest loop: `n` times (`k=2` to `n-1`)
  - Innermost two for loop together at most `e` times
  - Time complexity: `O(ne)`
- Note: if for one iteration of `k`. none of `dist[]` changes
  - Then it won't change for next iteration as well
  - Loop can be terminated with iterating till `n-1`

Summary

- Graph with negative edges
  – Dijkstra's algo does not work.
- Graph with negative cycles.
- Dynamic programming approach to graphs with negative cycles
  – Bellman Ford algorithm.