2020년 가을학기 알고리즘

Shortest Path

데이터네트워크연구실
문현수, 이영석
munhyunsu@cs-cnu.org

10주차 Feedback

- 제출률: 80%

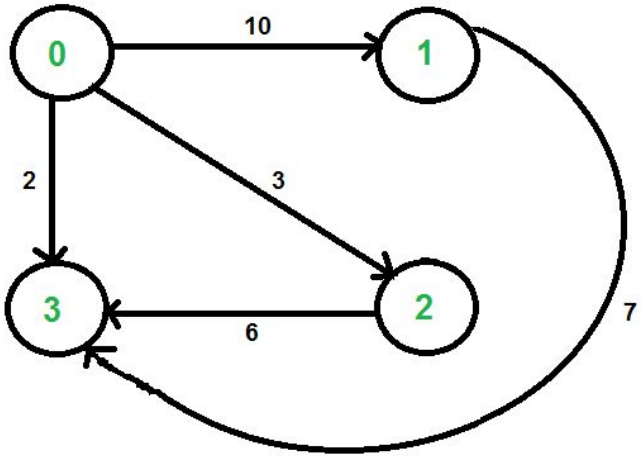
- 2020. 11. 21. (토) 학과 데브데이
  - 11. 12. (목) 기준 55% 참가

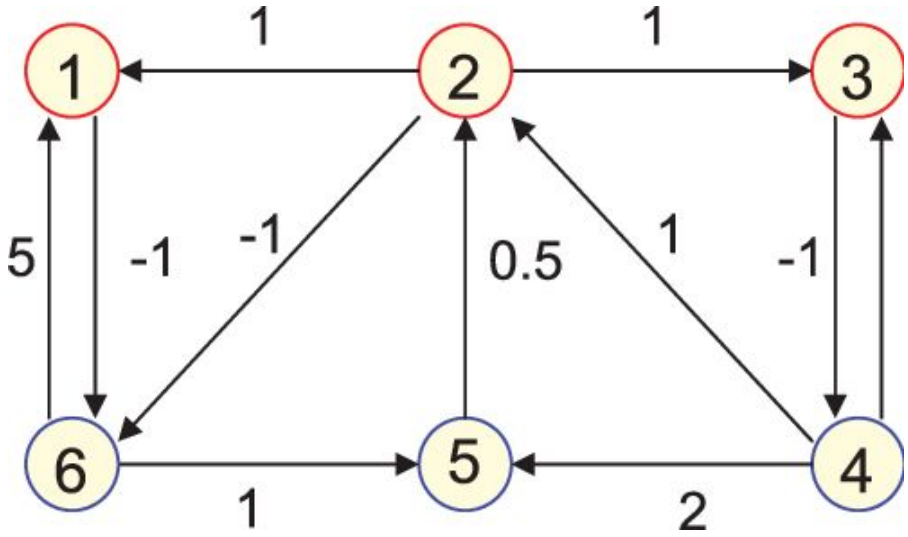
이번주 실습 목표

- 최단 거리
  - Weighted Graph
  - Dijkstra’s Algorithm
  - Bellman ford Algorithm

- 실습1: Dijkstra’s Algorithm (2점)
- 실습2: Bellman Ford Algorithm (3점)
- 실습3: Police Moving (5점)

Weighted Graph

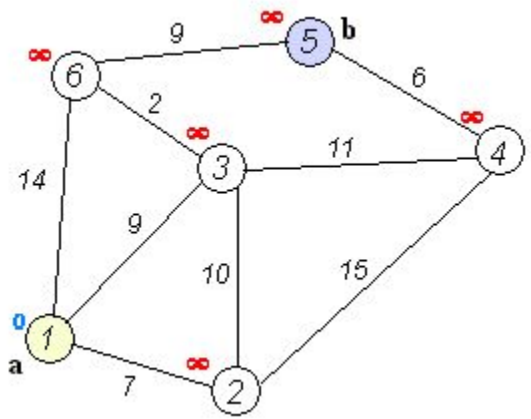




실습1. Dijkstra’s Algorithm (2)

Dijkstra’s Algorithm

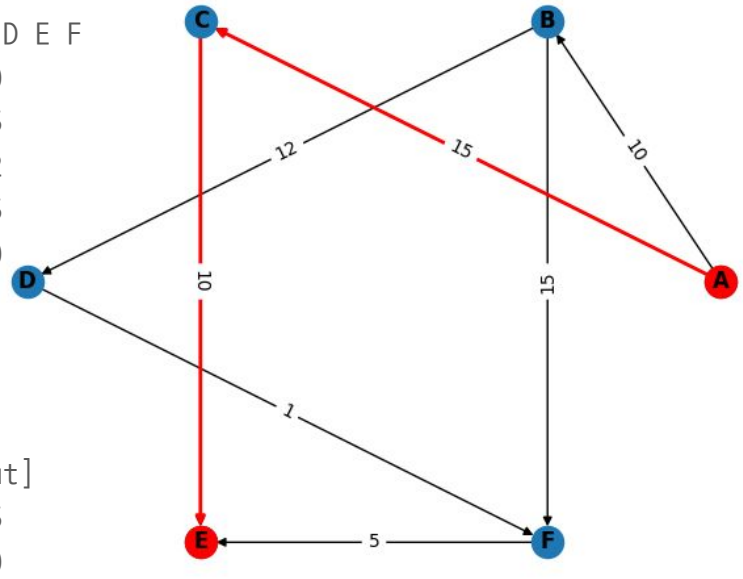
- 짧은 길 찾기 알고리즘 중 가장 유명한 알고리즘



Dijkstra’s Algorithm: 방향 가중치 그래프 연습

- 방향 가중치 그래프의 노드와 엣지가 주어졌을 때 방향 가중치 그래프를 그리고, 질의 노드 사이의 최단 거리 엣지 경로를 출력하시오.
  - 최단 거리는 1개가 보장된다.
  - 0 <= Weight <= 1,073,741,824

```
[Input]
6 7
A B C D E F
A B 10
A C 15
B D 12
B F 15
C E 10
D F 1
F E 5
A E

[Output]
A C 15
C E 10
```



Dijkstra’s Algorithm 수도코드

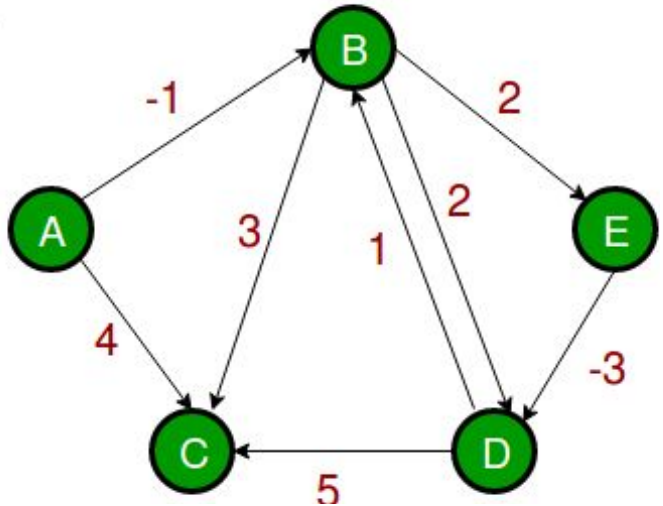
- 출처: [wikipedia]

```
1  function Dijkstra(Graph, source):
2
3      create vertex set Q
4
5      for each vertex v in Graph:             // Initialization
6          dist[v] ← INFINITY                  // Unknown distance from source to v
7          prev[v] ← UNDEFINED                 // Previous node in optimal path from source
8          add v to Q                          // All nodes initially in Q (unvisited nodes)
9
10     dist[source] ← 0                        // Distance from source to source
11
12     while Q is not empty:
13         u ← vertex in Q with min dist[u]    // Source node will be selected first
14         remove u from Q
15
16         for each neighbor v of u:           // where v is still in Q.
17             alt ← dist[u] + length(u, v)
18             if alt < dist[v]:               // A shorter path to v has been found
19                 dist[v] ← alt
20                 prev[v] ← u
21
22     return dist[], prev[]
```

실습2. Bellman Ford Algorithm (3)

Bellman-ford Algorithm

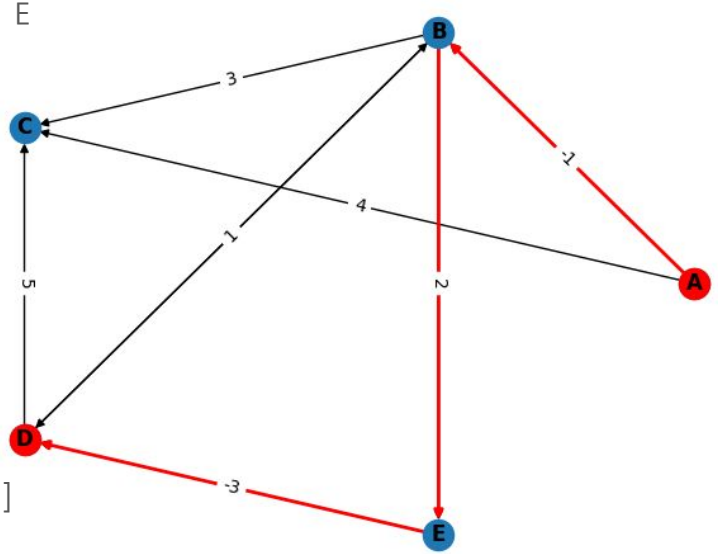
- 짧은 길 찾기 알고리즘 중 유명한 알고리즘
  - 음의 가중치를 고려할 수 있음!



Bellman-Ford Algorithm: 방향 가중치 그래프 연습

- 음의 가중치가 있는 방향 가중치 그래프의 노드와 엣지가 주어졌을 때 방향 가중치 그래프를 그리고, 질의 노드 사이의 **최단 거리 엣지 경로**를 출력하시오.
  만약, 무한 루프가 발생할 경우 Negative Cycle! 이라고 출력하시오.
  - 음의 가중치가 있음을 고려
  - 최단 경로는 1개 보장
  - Negative Cycle! (느낌표 포함!)
  - -1,073,741,824 <= Weight <= 1,073,741,824

```
[Input
5 8
A B C D E
A B -1
A C 4
B C 3
B D 2
B E 2
D B 1
D C 5
E D -3
A D

[Output]
A B -1
B E 2
E D -3
```

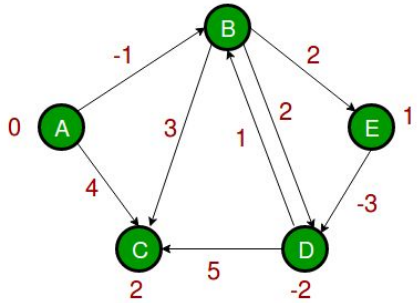


Bellman-ford 수도코드

- 출처: [Wikipedia]

```
function BellmanFord(list vertices, list edges, vertex source)
  ::distance[],predecessor[]

  // This implementation takes in a graph, represented as
  // lists of vertices and edges, and fills two arrays
  // (distance and predecessor) about the shortest path
  // from the source to each vertex

  // Step 1: initialize graph
  for each vertex v in vertices:
      distance[v] := inf             // Initialize the distance to all vertices to infinity
      predecessor[v] := null         // And having a null predecessor

  distance[source] := 0              // The distance from the source to itself is, of course, zero

  // Step 2: relax edges repeatedly
  for i from 1 to size(vertices)-1:
      for each edge (u, v) with weight w in edges:
          if distance[u] + w < distance[v]:
              distance[v] := distance[u] + w
              predecessor[v] := u

  // Step 3: check for negative-weight cycles
  for each edge (u, v) with weight w in edges:
      if distance[u] + w < distance[v]:
          error "Graph contains a negative-weight cycle"

  return distance[], predecessor[]
```

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	∞	1
0	-1	2	1	1
0	-1	2	-2	1

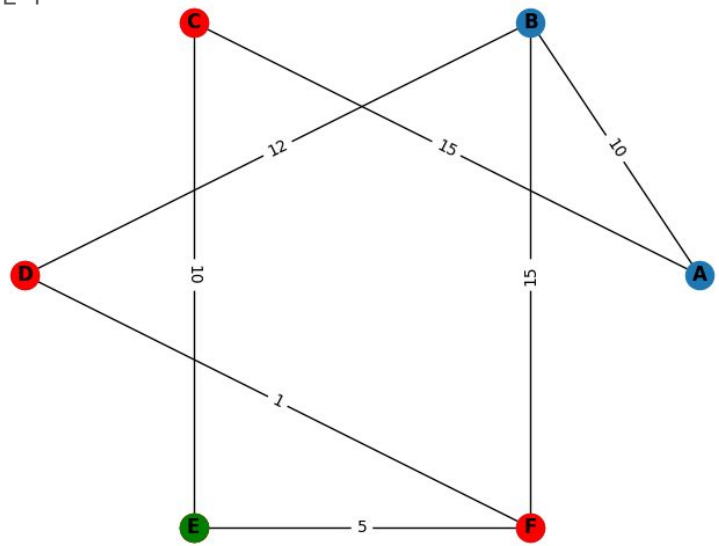


실습3. Police Moving (2)

경찰서 이전 계획

- 범죄 발생시 경찰 도착까지 오래 걸리는 문제를 해결하고자 경찰서의 위치를 변경하고자 한다. 마을과 마을간의 거리와 경찰이 쉽게 다다를 수 있는 거리가 입력으로 주어진다.
  가능한 많은 마을에 경찰이 쉽게 다다를 수 있는 곳에 경찰서를 이전하고자 한다. 어떤 마을에 경찰서를 설치해야하는지와 몇개의 마을에 쉽게 다다를 수 있는지 출력하시오.

```
[Input
6 7
A B C D E F
A B 10
A C 15
B D 12
B F 15
C E 10
D F 1
F E 5
10

[Output]
E 4
```



참고

Standard Library

- Python 3: https://docs.python.org/3/library/
- JAVA 11: https://cr.openjdk.java.net/~iris/se/11/latestSpec/api/
- C: https://en.cppreference.com/w/c
- CPP: https://en.cppreference.com/w/cpp

ID/ext	external ID	name
adb	ada	Ada
awk	awk	AWK
bash	bash	Bash shell
c	c	C
csharp	csharp	C#
cpp	cpp	C++
f95	f95	Fortran
hs	haskell	Haskell
java	java	Java
js	javascript	JavaScript
kt	kotlin	Kotlin
lua	lua	Lua
pas	pascal	Pascal
pl	pl	Perl
sh	sh	POSIX shell
plg	prolog	Prolog
py2	python2	Python 2
py3	python3	Python 3
r	r	R
rb	ruby	Ruby
scala	scala	Scala
swift	swift	Swift

주요 링크

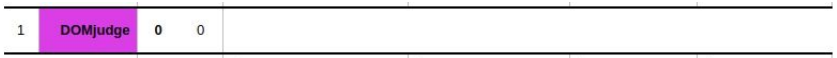
- 코딩테스트: http://coding.cnu.ac.kr:8080/domjudge/public
- FAQ : https://docs.google.com/document/d/1ntR6GS1SI7dRbYlw-pu8uT8U65Wc-RtMj10IEpiapfU/
- 질문: 메일!
- 문의 사항: munhyunsu@cs-cnu.org
  - ID / PASSWORD 변경
  - 실습 시간외 질문: [AL]질문제목

잊지 않아야 할 것) 소스코드 및 보고서

- 사이버 캠퍼스에 목요일까지 제출
  - 추가 시간 필요한 학생들도 목요일까지 제출. 추가 시간 문제 해결은 메일로도 제출!
- 보고서(.pdf 파일), 소스코드(.java, .py 등) zip 파일 압축
  - AL_학번_이름_06.zip (메일 추가 제출) or AL19_06.zip (사이버캠퍼스)
- 시간/공간 복잡도 해석(STL 고려), 자신의 생각, 질문, 느낀점, 공유하고 싶은 문제
  - 문제 해결을 위해 어떤 접근법을 사용하였는지, 무엇을 배웠고 느꼈는가?

보고서 템플릿: .pdf 로 제출!

- 알고리즘-x주차-주제
  학번 이름
- 코드 테스팅 결과 (점수표)



- 각 문제별 내용
  a. 문제 / 목표
  b. 해결 방법 (주요 소스코드 첨부. 스크린샷 or 코드 CV)
  c. 결과 (입력, 출력 결과)
- 느낀점: 과제를 하며 느낀 점 / 공유하고싶은 문제 / 난이도 / 부탁 / 조교에게... 등