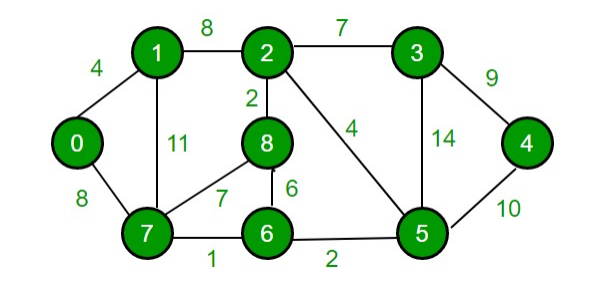
1. 소개

1-1. concept of Greedy algorithms

For many optimization problems, using dynamic programming to determine the best choices is overkill; simpler, more efficient algorithms will do.

A greedy algorithm always makes the choice that looks best at the moment. That is, it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution. (출처p414)

The choice made by a greedy algorithm may depend on choices so far, but it cannot depend on any future choices or on the solutions to subproblems. Thus, unlike dynamic programming, which solves the subproblems before making the first choice, a greedy algorithm makes its first choice before solving any subproblems. A dynamic programming algorithm proceeds bottom up, whereas a greedy strategy usually progresses in a top-down fashion, making one greedy choice after another, reducing each given problem instance to a smaller one. (출처 : p424)



1-2. 그리디 알고리즘을 적용한 실생활 예시 1개

You go to a shop and buy an ice-cream and it costs 700won. you paid 10000won for it. you get your change one by one, never exceeding the total change.

|  |  |  |  |
| --- | --- | --- | --- |
| step | What the cashier gave you | How much left | Total money you get |
| 0 | 0 | 9300 | 0 |
| 1 | 5000 | 4300 | 5000 |
| 2 | 1000 | 3300 | 6000 |
| 3 | 1000 | 2300 | 7000 |
| 4 | 1000 | 1300 | 8000 |
| 5 | 1000 | 300 | 9000 |
| 6 | 100 | 200 | 9100 |
| 7 | 100 | 100 | 9200 |
| 8 | 100 | 0 | 9300 |

We didn’t violate the 2 rules and we took the best choices on every step.

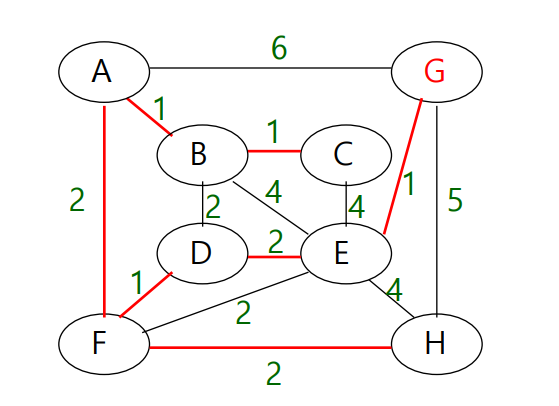
(순서대로 간단히 설명 5000, 1000, 1000, 1000, 1000, 100, 100, 100원)

1-3. 그리디 알고리즘을 적용한 알고리즘 예시

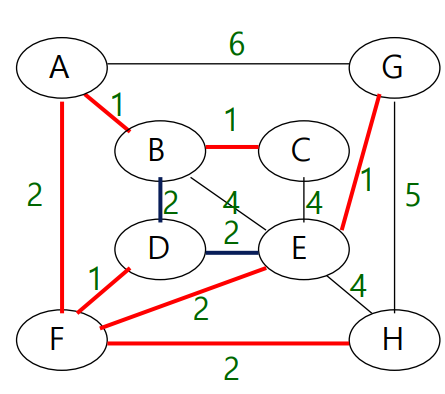
Prim’s Algorithm:Solution

Kruskal’s Algorithm

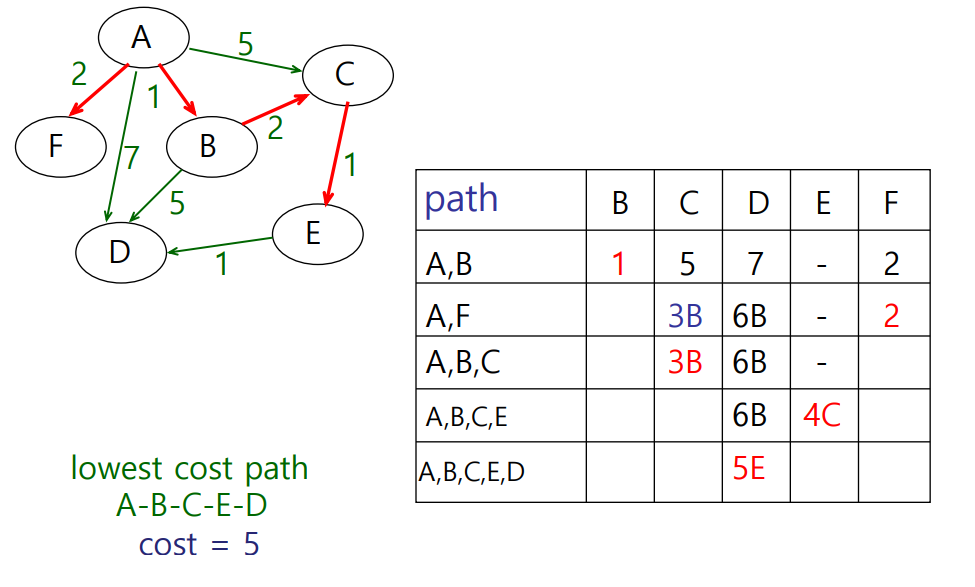
Dijkstra’s algorithm



Prim’s algorithm



Kruskal’s algorithm



Dijkstra’s Algorithm

(세세하게 설명하지 않고 A에서 G까지 갈 때 위 Prim’s algorithm을 이용하면 이 경로로 도달하고 Kruskal’s algorithm를 이용하여 이 경로로 G에 도달한다고 설명, dijkstra’s algorithm에서는 A노드에서 시작하여 e 노드에 도달)

2. 세부 사항

2-1. 그리디 알고리즘의 특징

As we mentioned before, A greedy algorithm always makes the choice that looks best at the moment. So Greedy algorithms do not always yield optimal solutions, but for many problems they do.

2-2. 그리디 알고리즘의 적용(탐욕적 선택 속성, 최적 부분 구조)

* Greedy-choice property

The first key ingredient is the greedy-choice property: we can assemble a globally optimal solution by making locally optimal (greedy) choices. In other words, when we are considering which choice to make, we make the choice that looks best in the current problem, without considering results from subproblems (출처 : p424)

* Optimal substructure

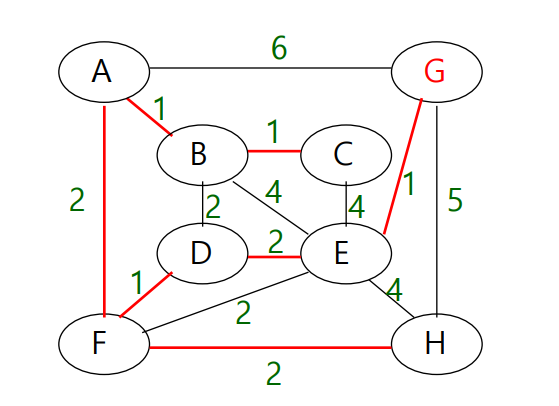
optimal solutions to a problem incorporate optimal solutions to related subproblems, which we may solve independently. (출처 : p362)

(설명할 때 : 주어진 문제가 부분문제를 풀기 위해 모든 가능한 수단을 쓰는 것이 아니라 부분문제들을 풀기 위해 optimal solution을 사용함으로써 풀어진다)

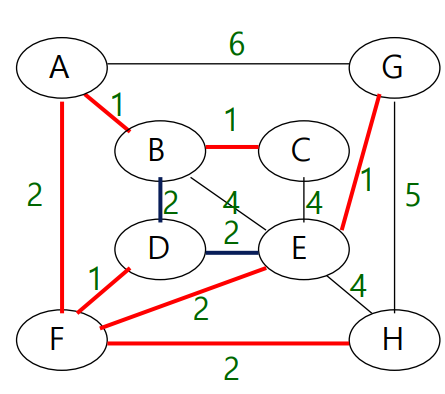
2-3. 그리디 알고리즘의 장점

Using dynamic programming to determine the best choices is overkill if a greedy algorithm works since greedy algorithm is usually simpler and faster. Even though greedy algorithms don’t always provide us with optimal solutions, they’re often used to find good approximations.

2-5. 1-3에서 소개한 예시 등에서 그리디 알고리즘이 최적해를 찾는 경우/찾지 못하는 경우 (단점)

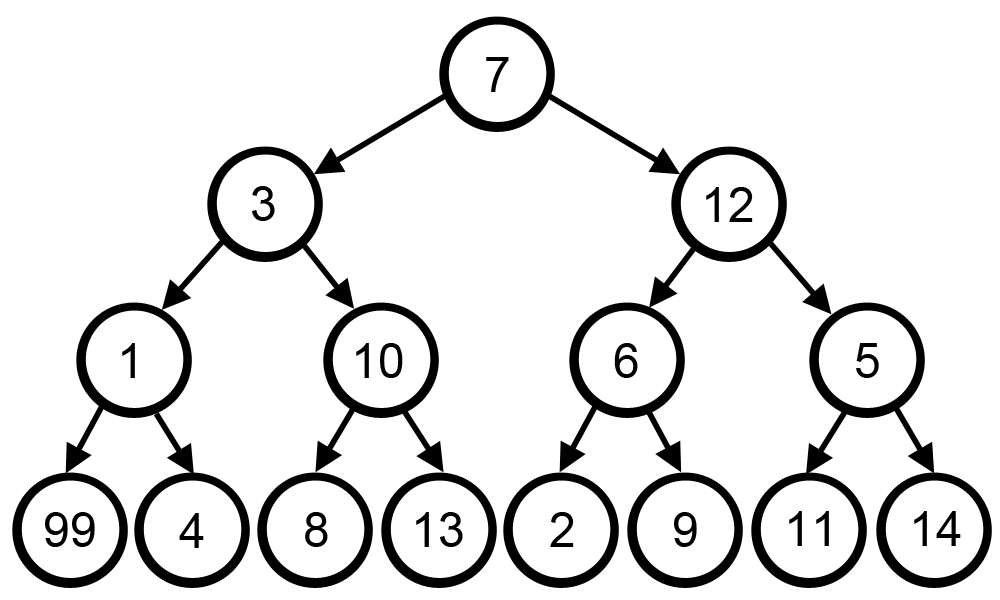


Prim’s algorithm



Kruskal’s algorithm

In this case, Prim’s algorithm didn’t find the optimal solution, but Kruskal’s algorithm found it. This doesn’t Kruskal’s algorithm is better than Prim’s algorithm but Greedy algorithms could yield nonoptimal solutions.



(발표할때 : Root node에서 leaf node로 갈 때 거치는 노드 값들의 합이 제일 큰 노드들을 찾는다. Greedy algorithm을 이용하면 이 경우에서 optimal solution을 가지지 못한다. 이 경우에서 7, 3, 1, 99로 가는 것이 제일 합이 크기 때문에 optimal solution을 찾지 못하였다.)

|  |
| --- |
| 김원, 「Data Structures: Graphs: Traversal, Minimum Spanning Tree, Graph Traversal Algorithms」, Gachon university, p58-p87  Thomas H. Cormen, 「Introduction to Algorithms, 3rd Edition」, The MIT Press, 2009, p.362-p424 |