파이썬으로 배우는 알고리즘 기초

Chap 4. 탐욕 알고리즘

4.1 서로소 집합과 크루스칼 알고리즘

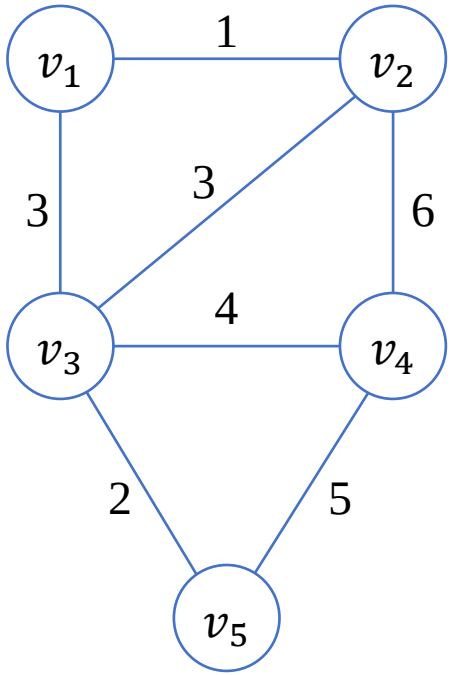
4.1 최소비용 신장트리

- 최소비용 신장트리: 크루스칼 알고리즘(Kruskal's Algorithm)
  - 1단계(초기화): 해답의 집합을 공집합으로 둔다.$F = \phi$
    - V의 서로소 집합(disjoint set)을 생성한다.
    - E를 비내림차순으로 정렬한다.
  - 2단계(선택): 최적의 원소 하나를 해답의 집합에 포함시킨다.
    - 정렬된 E 집합에서 간선 $e = (i, j)$를 선택
    - 두 정점 i, j가 속한 집합 p, q를 찾아서 (**Find**),
      p, q 가 같으면 e를 버리고, 다르면 F에 e를 포함한 후, p, q를 합친다 (**Union**).
  - 3단계(검사): $|F| = n - 1$이면 종료, 아니면 2단계를 반복한다.

4.1 최소비용 신장트리

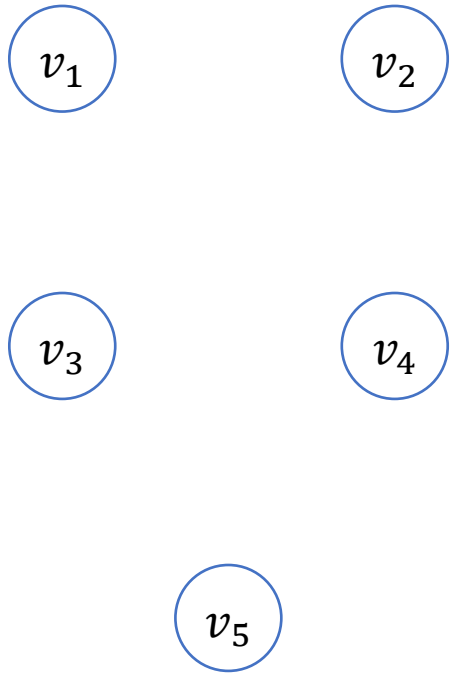
- Determine a minimum spanning tree.

1. Edges are sorted by their weights



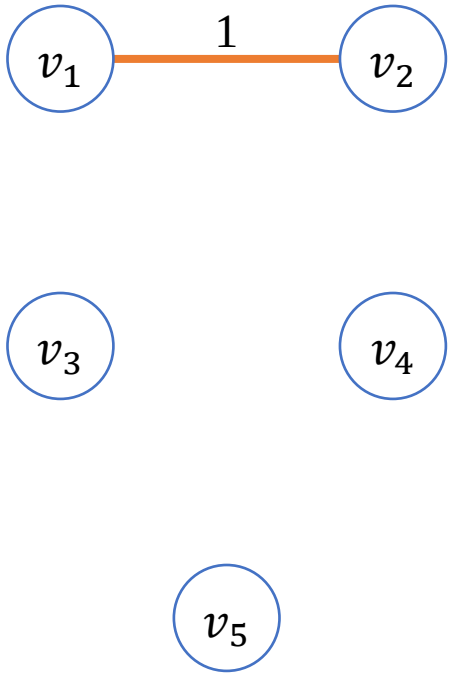
edges	*weight*
(v_1, v_2)	1
(v_3, v_5)	2
(v_1, v_3)	3
(v_2, v_3)	3
(v_3, v_4)	4
(v_4, v_5)	5
(v_2, v_4)	6

2. Disjoint sets are created

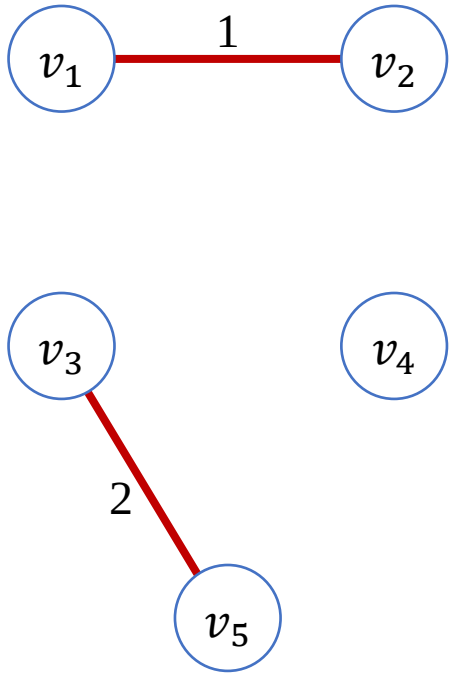


4.1 최소비용 신장트리

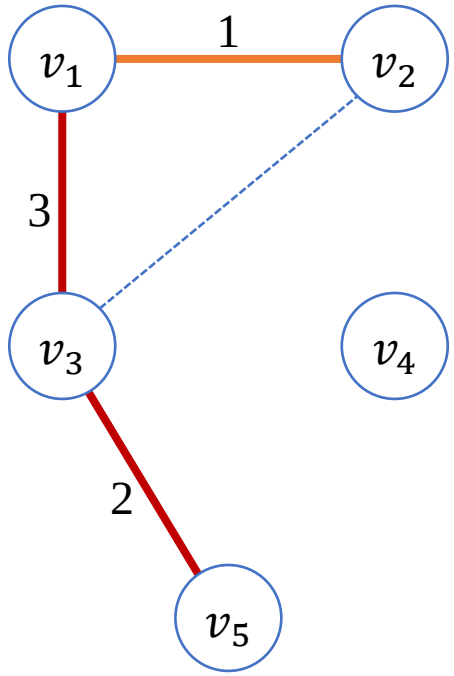
3. The first edge (v_1, v_2) is selected



4. Next edge (v_3, v_5) is selected



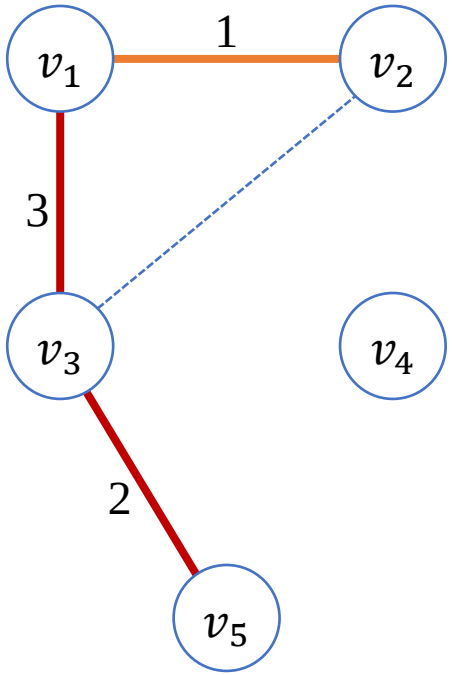
5. Next edge (v_1, v_3) is selected



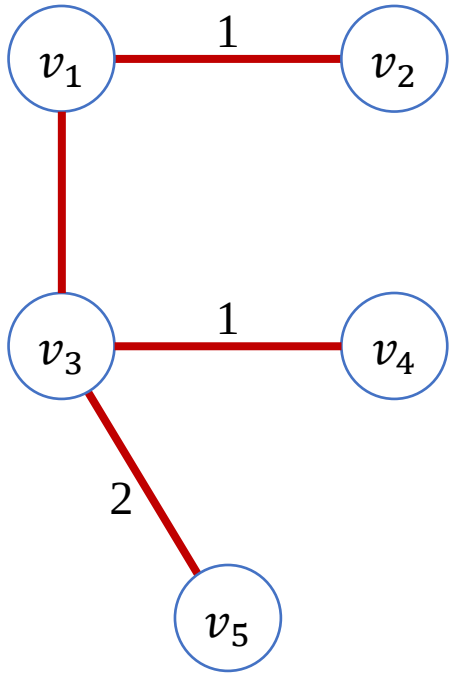
edges	*weight*
(v_1, v_2)	1
(v_3, v_5)	2
(v_1, v_3)	3
(v_2, v_3)	3
(v_3, v_4)	4
(v_4, v_5)	5
(v_2, v_4)	6

4.1 최소비용 신장트리

6. Next edge (v_2, v_3) is discared, because it creates a cycle



7. Next edge (v_3, v_4) is selected



- (v_4, v_5) is not considered, because all the subsets are merged

edges	*weight*
(v_1, v_2)	1
(v_3, v_5)	2
(v_1, v_3)	3
(v_2, v_3)	3
(v_3, v_4)	4
(v_4, v_5)	5
(v_2, v_4)	6

4.1 최소비용 신장트리

- **사이클 탐지**를 어떻게 하지?
  - 서로소 집합 (Disjoint Set)
    - 교집합이 공집합인 두 집합 A, B는 서로소 집합. $A \cap B = \emptyset$
  - Union-Find 알고리즘
    - 서로소 집합 자료구조를 이용해서
    - 두 개의 원소가 같은 집합에 속하는 지를 판단할 수 있는 알고리즘

4.1 최소비용 신장트리

- **전체집합** $U = \{ A, B, C, D, E \}$

```
for i in U:
    makeset(i);
```

$\{A\}$ $\{B\}$ $\{C\}$ $\{D\}$ $\{E\}$ **(disjoint sets)**

```
p = find(B);
q = find(C);
```

p q

```
merge(p, q);
```

$\{A\}$ $\{B, C\}$ $\{D\}$ $\{E\}$

```
p = find(C);
q = find(E);
```

p q

```
equal(C, E);
```
returns false;

```
merge(p, q);
```

$\{A\}$ $\{B, C, E\}$ $\{D\}$

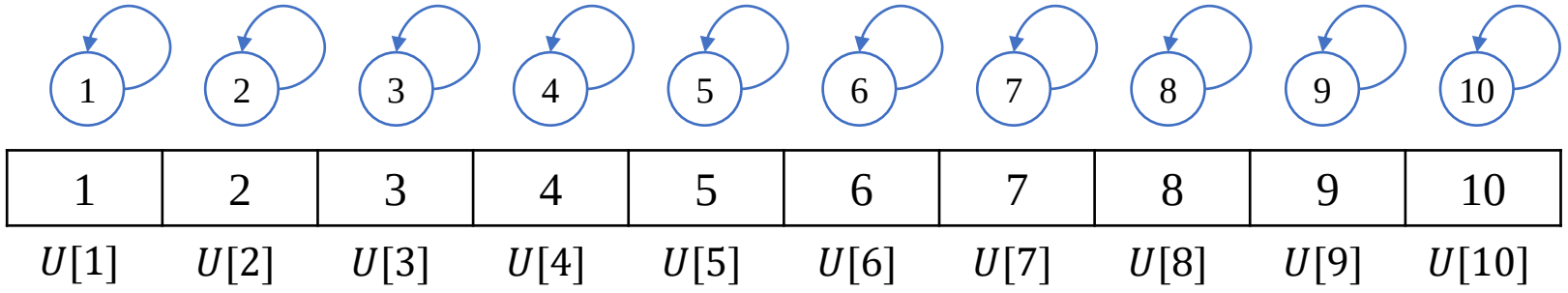
```
p = find(C);
q = find(E);
```

p q

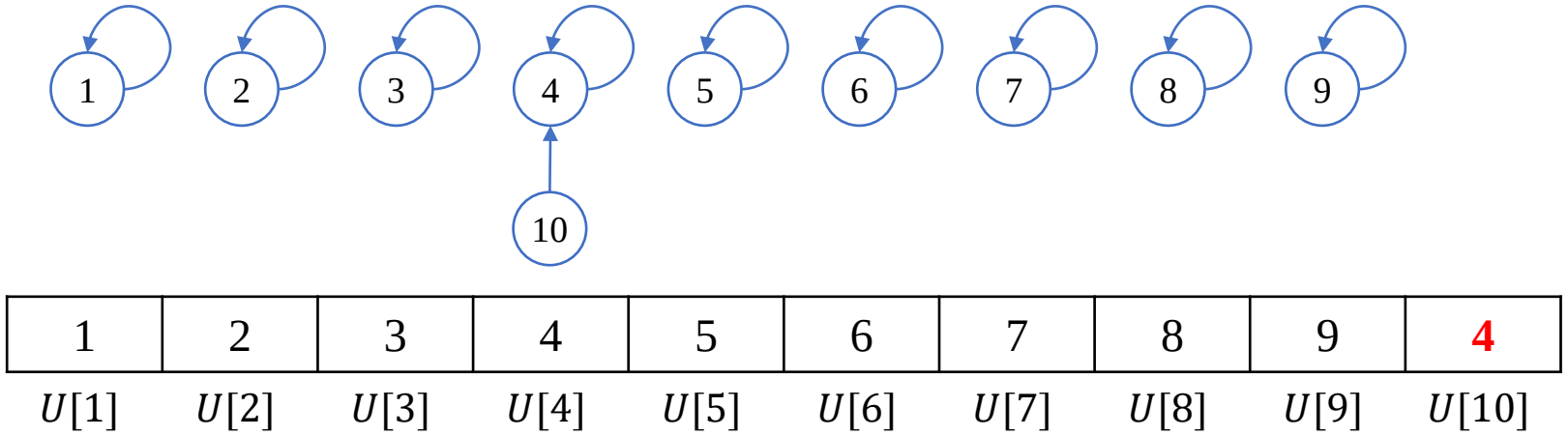
```
equal(C, E);
```
returns true;

4.1 최소비용 신장트리

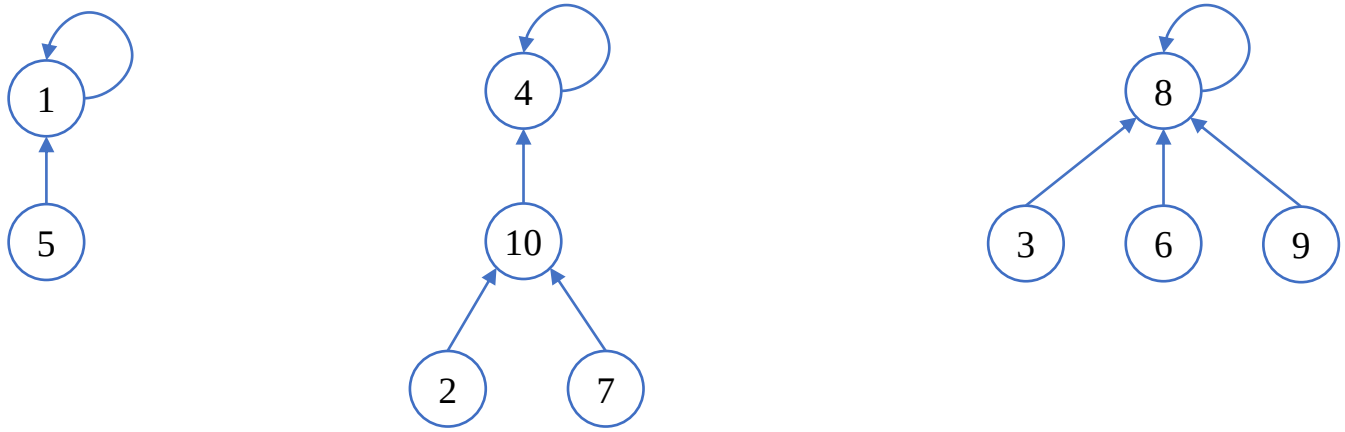

```
init(10);
```




```
merge(find(4), find(10);
```



4.1 최소비용 신장트리



1	10	8	4	1	8	10	8	8	4
$U[1]$	$U[2]$	$U[3]$	$U[4]$	$U[5]$	$U[6]$	$U[7]$	$U[8]$	$U[9]$	$U[10]$

4.1 최소비용 신장트리

Algorithm 4.2: Disjoint Set: Union-Find

```
class DisjointSet:

    def __init__ (self, n):
        self.U = []
        for i in range(n):
            self.U.append(i)

    def equal (self, p, q):
        if (p == q):
            return True
        else:
            return False

    def find (self, i):
        j = i
        while (self.U[j] != j):
            j = self.U[j]
        return j

    def union (self, p, q):
        if (p < q):
            self.U[q] = p
        else:
            self.U[p] = q
```

Algorithm 4.2: Kruskal's Algorithm

```
def kruskal (n, E):
    F = []
    dset = DisjointSet(n)
    while (len(F) < n - 1):
        edge = E.pop(0)
        i, j = edge[0], edge[1]
        p = dset.find(i)
        q = dset.find(j)
        if (not dset.equal(p, q)):
            dset.union(p, q)
            F.append(edge)
    return F
```

4.1 최소비용 신장트리


```
INF = 999
n = 5
E = [
    [0, 1, 1],
    [2, 4, 2],
    [0, 2, 3],
    [1, 2, 3],
    [2, 3, 4],
    [3, 4, 5],
    [1, 3, 6],
]

F = kruskal(n, E)
for i in range(len(F)):
    print(F[i])

print(cost(F))
```

주니온TV@Youtube

자세히 보면 유익한 코딩 채널

https://bit.ly/2JXXGqz

- 여러분의 **구독**과 **좋아요**는 강의제작에 큰 힘이 됩니다.
- 강의자료 및 소스코드: **구글 드라이브**에서 다운로드
  (다운로드 주소는 영상 하단 설명란 참고)

  https://bit.ly/3fN0q8t