1197 최소 스패닝 트리.

- 문제 : · 정점의 개수 ($1 \le V \le 10000$)
  · 간선의 개수 ($1 \le E \le 100000$)
  · 무방향 그래프
  · 트리의 가중치는 정수 범위.
  → 모든 정점 연결하는 최소 가중치 구하기.

- 구현 ① 프림 (68 ms)
  · 정점의 확장.
  · 임의의 정점을 (1개) 선택하고 시작.
  · 선택된 정점 ——— 선택 X 정점들 간의 최소 비용을 갖는 정점으로 연결

  ⇒ 각 단계에서 연결 후보 몇 개 중에서 최소값을 찾기 위해서는 "힙"으로 구현하면 매번 $O(\log E)$만큼만에 최소값 찾을 수 있다.

  1. 모든 간선에 대한 정보를 각 정점에 연결된 다른 정점, 그리고 해당 가중치(비용)를 기준으로 벡터에 저장해둠.
  2. 임의의 한 정점을 선택하고 최소힙에 (가중치, 정점(후보)) 순서의 쌍으로 넣는다.

  반복 {
  3-1) 최소힙에서 꺼내고 방문여부 체크한다.
  < 방문 O : pass (중복이 문제되지 않는 이유).
  X : 방문체크하고 결과 값에 합산한다.
  3-2) 선택된 정점에 연결된 간선정보를 (트리 가중치) 힙에 추가한다.

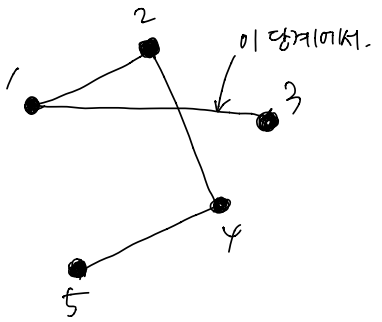
* min_heap은
① priority_queue<type, vector<type>, greater<type>> q;
이렇게 선언된다. (항상 less가 기본값. 여기서는 max_heap).
② max_heap으로 구현하고 음수를 취한 값을 넣으면 반대로 정렬됨.
(꺼내서 쓸 때는 다시 음수 취해야 함).

구현 ② 크루스칼 (36ms).

- 간선의 확장.
- 두 정점을 잇고 가중치를 갖는 간선에 대한 정보를 담는 벡터에 모든 간선 정보를 넣고 시작.
- 가중치 기준으로 간선 벡터를 정렬하고 시작.
- 가장 가중치가 작은 간선을 선택하되, 같은 집합에 속하는 것은 PASS.
  => 같은 집합 여부를 확인하기 위해 간선이 선택되었을 때 결정된 루트 노드를 찾는 Find 함수 필요.

  ↳ 그러기 위해서,
  1. 처음에 자기 자신을 루트로하는 parent 배열을 만든다.
  2. 간선이 선택될 때 (같은집합이 X)
     start 또는 end 정점을 루트로 지정하도록 한다.

  * 그렇게 되면 최종적으로 다른 노드를 루트로 하는 노드는 루트노드가 아닐것이며, 자기 자신을 루트로 하는 루트노드를 하나씩 재귀로 따라가며 찾을수 있을 것이다.




```
P[1] = 5
P[2] = 1
P[3] = 1.
P[4] = 5
P[5] = 5.
```

만약 중간 노드가 루트가 아닌 다른 노드를 루트로 가리키고있다고 하더라도 다시 그 노드의 루트 ... 를 따라가면 결국 자신을 가리키는 전짜 루트를 찾을수 있다. (start, end 순서가 상관 없는 이유!!)

3. Find 함수를 통해 후보의 양 정점의 루트를 비교해서 다르다면 한쪽의 루트를 다른쪽의 루트가 루트로 지정하도록 parent 배열값을 변경해서 같은 집합으로 만들어준다. => Union 함수.