6 Priority Queues and Disjoint Sets

6.1 Computing k-th smallest element in a heap

Input. An array $A[1..n]$ of integers satisfying the min-heap property (i.e., for all $2 \le i \le n$, $A[\lfloor i/2 \rfloor] \le A[i]$) and an integer $1 \le k \le n$.

Output. The k-th smallest element in A.

Running time. $O(k \log k)$.

Remark. A naive way to do this is to call ExtractMin k times. This however will result in $O(k \log n)$ time instead of $O(k \log k)$. The running time $O(k \log k)$ guarantees that for any constant k (i.e., $k = O(1)$) we find the k-th smallest element in constant time!

Sample.

Input. $A = [2, 10, 7, 11, 12, 15, 8, 30, 72, 14]$, $k = 4$.

Output. 10.

7 Decomposition of graphs

7.1 Computing sums of degrees of neighbors

Input. A list of edges of an *undirected* graph $G(V, E)$. Assume that $V = \{1, 2, \ldots, n\}$.

Output. For each vertex $v \in V$, compute the sum of degrees of v's neighbors.

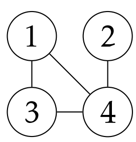
Running time. $O(|V| + |E|)$.

Sample.

Input. $\{1, 4\}$, $\{1, 3\}$, $\{3, 4\}$, $\{4, 2\}$.

Output. $5, 3, 5, 5$.

Explanation. The degrees of the vertices 1, 2, 3, 4 are 2, 1, 2, 3, respectively. The sum of degrees of the neighbors of vertex 1 is $2 + 3 = 5$, the sum of degrees of the neighbors of vertex 4 is $2 + 1 + 2 = 5$.



7.2 Checking hamiltonicity of a DAG

Description. A path in a graph is called *Hamiltonian* if it visits each vertex of the graph exactly once.

Input. A list of edges of a *directed acyclic* graph $G(V,E)$. Assume that $V = \{1,2,\ldots,n\}$.

Output. Output the sequence of vertices of a Hamiltonian path if it exists, or “no path” if there is no such path.

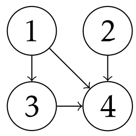
Running time. $O(|V|+|E|)$.

Sample 1.

Input. $(1,4)$, $(1,3)$, $(3,4)$, $(2,4)$.

Output. No path.

Explanation.

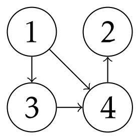


Sample 2.

Input. $(1,4)$, $(1,3)$, $(3,4)$, $(4,2)$.

Output. 1, 3, 4, 2.

Explanation.

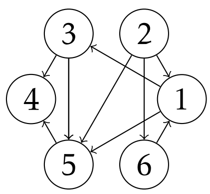


Sample 3.

Input. $(3,5)$, $(6,1)$, $(1,5)$, $(2,6)$, $(2,1)$, $(1,3)$, $(5,4)$, $(3,5)$, $(2,5)$, $(3,4)$.

Output. 2, 6, 1, 3, 5, 4.

Explanation.



8 Paths in graphs

8.1 Finding a shortest cycle

Input. A directed graph $G(V,E)$ with positive edge lengths and an edge $e \in E$.

Output. The length of the shortest cycle containing the edge e, or "no cycle".

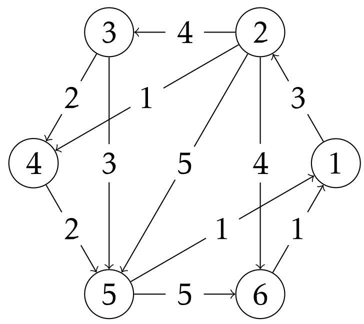
Running time. $O(|V|^2)$.

Sample.

Input. E: (1,2,3), (2,3,4), (3,4,2), (4,5,2), (5,6,5), (6,1,1), (2,4,1), (3,5,3), (2,5,5), (5,1,1), (2,6,4); $e = (1,2)$.

Output. 7.

Explanation. The shortest cycle through the edge $(1,2)$ is $1 \to 2 \to 4 \to 5 \to 1$.



8.2 Generalized shortest paths

Description. In Internet routing, there are delays on lines but also, more significantly, delays at routers. This motivates a generalized shortest-paths problem.

Suppose that in addition to having edge lengths $\{l_e : e \in E\}$, a graph also has *vertex costs* $\{c_v : v \in V\}$. Now define the cost of a path to be the sum of its edge lengths, *plus* the costs of all vertices on the path (including the endpoints).

Input. A directed graph $G(V,E)$ with positive edge costs and positive vertex costs, a starting vertex $s \in V$.

Output. An array $cost[\cdot]$ such that for every vertex u, $cost[u]$ is the least cost of any path from s to u (i.e., the cost of the cheapest path), under the definition above. (In particular, $cost[s] = c_s$.)

Running time. $O(|V|^2)$.