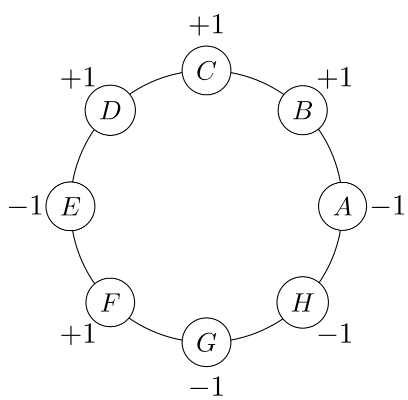
CSE421: Design and Analysis of Algorithms	May 5th, 2021
Homework 5	
Shayan Oveis Gharan	Due: May 13, 2021 at 23:59 PM

P1) Suppose we have $2n$ points around a circle for some integer $n \geq 1$, where n of them are labelled with $+1$ and n of them are labelled with -1. Use induction to prove that you can pick one of the $2n$ points as your starting point, then move *counter-clockwise* around the circle such that at any point the number of $+1$ points you have passed through is at least the number of -1 points.

For example, in the following picture if you start at D you get the following values when you move counter clockwise. For example, at D you just have seen one $+1$ and no -1, at E you have seen one $+1$'s and one -1, when you get to H you have seen two $+1$'s and three -1's. So D is not a good starting point because you will get negative at H.

	D	E	F	G	H
Sum	+1	0	+1	0	-1



On the other hand,, if you start at B, you will always have a non-negative sum at all points. So, B is the answer in this case.

B	C	D	E	F	G	H	A
+1	+2	+3	+2	+3	+2	+1	0

P2) Given a connected graph $G = (V, E)$ with n vertices and m edges where every edge has a positive weight $w_e > 0$, for any pair of vertices u, v define $d(u, v)$ to denote the length of the shortest path from u to v in G.

a) Prove that $d(., .)$ is a metric, namely it satisfies the following three properties: (i) $d(u, v) \geq 0$ for all u, v and $d(u, v) = 0$ only when $u = v$. (ii) $d(u, v) = d(v, u)$ for all vertices $u, v \in V$. (iii) $d(u, v) + d(v, w) \geq d(u, w)$ for all $u, v, w \in V$.

b) Let $d^* := \max_{u,v \in V} d(u,v)$ denote the longest shortest path in G. Design an $O(m \log(n))$ time algorithm that gives a 2-approximation to d^*, i.e., your algorithm should output a number $\tilde{d}^*$ such that

$$\tilde{d}^* \leq d^* \leq 2\tilde{d}^*.$$

P3) In the *Hamiltonian Path* problem, you are given an unweighted undirected connected graph $G = (V, E)$ with n vertices together with two vertices s, t and you need to output a *path* from s to t of length $n - 1$, i.e., a path that starts at s goes to all vertices and ends at t, or output "Impossible" if no such path exists. Recall that the Hamiltonian path problem is NP-complete. Suppose a friend of yours is came up with an efficient algorithm to solve this problem. Unfortunately, their code does not output the Hamiltonian path; Instead, for any graph G and *any* pair of vertices s, t, if G has such a path from s to t it will output "yes" and "no" otherwise. Now, given a graph G with n vertices and s, t, design a polynomial time algorithm (that only runs their code polynomially many times) and outputs a Hamiltonian path from s to t in G if it exists, and outputs "Impossible" otherwise.

P4) Draw the dynamic programming table of the following instance of the knapsack problem: You are given 6 items with weight $1, 2, 4, 6, 7, 9$ and value $1, 3, 6, 11, 18, 24$ respectively and the size of your knapsack is 13.

P5) **Extra Credit:** A k-hypergraph is composed of a set V of vertices and a set of hyperedges where every hyperedge is a subset of V of size at least 2 and at most k, i.e., S is a hyperedge if $S \subseteq V$ and $2 \leq |S| \leq k$. Note that 2-hypergraph is the same as a graph. Given a k-hypergraph $G = (V, E)$ with n vertices where for some $k \geq 2$ design a k-approximation algorithm for the vertex cover problem: Find the minimum set W of vertices of G such that every hyperedge $S \in E$ has at least one vertex of W.