Basic Algorithms, Assignment 9
Solutions

1. Let $a(x) = \sum_{j<n} a_j x^j$ be a polynomial of degree less than n. Find $a(0)$ as a simple expression of $a(1), a(\epsilon), a(\epsilon^2), \ldots, a(\epsilon^{n-1})$ where $\epsilon = e^{2\pi i/n} = \cos(2\pi/n) + i\sin(2\pi/n)$. (Idea: Inverse DFT)
   `Solution:` As $a(0) = a_0$, the constant term, we apply the inverse DFT to $a(1), a(\epsilon), a(\epsilon^2), \ldots, a(\epsilon^{n-1})$ – this is the case where we have n equations in n unknowns and adding them all up gives n times the zero-th unknown so we get the nice formula
   $$a(0) = \frac{1}{n}\sum_{j=0}^{n-1} a(\epsilon^j)$$
   That is, the value at 0 is the average on the n values surrounding it. FYI: This is a fundamental result when studying functions of a complex variable. As $n \to \infty$ the average (often!) approaches an integral and $a(0)$ (often!) gets well approximated by its Taylor Series to n terms so we get (caution: not always!!)
   $$a(0) = \frac{1}{2\pi}\int_0^{2\pi} a(e^{2\pi i\theta})d\theta$$

2. Consider the undirected graph with vertices $1, 2, 3, 4, 5$ and adjacency lists (arrows omitted) $1 : 25$, $2 : 1534$, $3 : 24$, $4 : 253$, $5 : 412$. Show the d and π values that result from running BFS, using 3 as a source. Nice picture, please!
   `Solution:`
   BFS: $3, 2, 4, 1, 5$
   $d[3] = 0$, $\pi[3] = nil$
   $d[2] = 1$, $\pi[2] = 3$
   $d[4] = 1$, $\pi[4] = 3$
   $d[1] = 2$, $\pi[1] = 2$
   $d[5] = 2$, $\pi[5] = 2$

3. Show the d and π values that result from running BFS on the undirected graph of Figure A, using vertex u as the source.
   `Solution:`
   $d[U] = 0$, $\pi[U] = nil$
   $d[T] = 1$, $\pi[T] = U$
   $d[X] = 1$, $\pi[X] = U$

$d[Y] = 1$, $\pi[Y] = U$
$d[W] = 2$, $\pi[W] = T$
$d[S] = 3$, $\pi[S] = W$
$d[R] = 4$, $\pi[R] = S$
$d[V] = 5$, $\pi[V] = R$

4. We are given a set V of wrestlers. Between any two pairs of wrestlers there may or may not be a rivalry. Assume the rivalries form a graph G which is given by an adjacency list representation, that is, $Adj[v]$ is a list of the rivals of v. Let n be the number of wrestlers (or nodes) and r the number of rivalries (or edges). Give a $O(n + r)$ time algorithm that determines whether it is possible to designate some of wrestles as `GOOD` and the others as `BAD` such that each rivalry is between a `GOOD` wrestler and a `BAD` wrestler. If it is possible to perform such a designation your algorithm should produce it.

   Here is the approach: Create a new field `TYPE`$[v]$ with the values `GOOD` and `BAD`. Assume that the wrestlers are in a list L so that you can program: For all $v \in L$. The idea will be to apply `BFS[v]` – when you hit a new vertex its value will be determined. A cautionary note: `BFS[v]` might not hit all the vertices so, just like we had `DFS` and `DFS-VISIT` you should have an overall `BFS-MASTER` (that will run through the list L) and, when appropriate, call `BFS[v]`.

   `Note:` The cognescenti will recognize that we are determining if a graph is bipartite!
   `Solution:`The idea here is to call the first wrestler `GOOD`. When someone is adjacent to someone `GOOD` they are called `BAD` and if they are adjacent to someone `BAD` they are called `GOOD`. But if in the adjacency list you come upon someone who has already been labelled (that is, not white) then you must check if there is a contradiction. A further problem: `BFS[v]` will only explore the connected component of v, if that is labelled with no contradiction then you must go on to the other vertices. So we start with everything white. The "outside" program is:
   For all $v \in L$
   If $COLOR[v] = WHITE$ (*else skip*) then `BFSPLUS[v]`.

   `BFSPLUS[v]` starts by setting `TYPE`$[v] = GOOD$. Then it runs `BFS[v]` with two additions. When $u \in Adj[w]$ and u is white you define `TYPE`$[u]$ to be the opposite of `TYPE`$[w]$. When u is not white you check if `TYPE`$[w] =$ `TYPE`$[u]$. If not, ignore. But if so exit the entire program with `NO DESIGNATION POSSIBLE` printout.

5. Show how DFS works on Figure B. All lists are alphabetical, except that we put R before Q so it is the first letter. Show the discovery and finishing time for each vertex.
   `Solution:`
   $Discovery\ order : RUYQSVWTXZ$
   $Finishing\ order : WVSZXTQYUR$
   $Stack : push(R)\ push(U)\ push(Y)\ push(Q)\ push(S)\ push(V)\ push(W)$
   $pop(W)\ pop(V)\ pop(S)\ push(T)\ push(X)\ push(Z)\ pop(Z)$
   $pop(X)\ pop(T)\ pop(Q)\ pop(Y)\ pop(U)\ pop(R)$

6. Show the ordering of the vertices produced by `TOP-SORT` when it is run on Figure C, with all lists alphabetical.
   `Solution:`We apply `DFS` to the graph. The first letter is M so we apply `DFS-VISIT(M)`

v	s[v]	f[v]
M	1	20
Q	2	5
T	3	4
R	6	19
U	7	8
Y	9	18
V	10	17
W	11	14
Z	12	13
X	15	16

Note, for example, that though X is in Adj[M] it doesn't affect DFS. At time 19 R finishes and returns control to M. M looks at X in its adjacency list but it is no longer white and so ignores it. At this stage all vertices are black except N, O, P, S which as white. In this particular example N is the letter right after M but in the general case DFS would skip over those vertices which weren't white. Indeed, right after DFS-VISIT all vertices are white or black. So next we do `DFS-VISIT(N)`. Note that the time does *not* restart! Note also that the now black vertices, such as $U \in Adj(N)$ and $R \in Adj(O)$, do not play a role

v	s[v]	f[v]
N	21	26
O	22	25
S	23	24

Finally we do `DFS-VISIT(P)`. This one is quick. The adjacency list of P consists only of S which is already black. So

v	s[v]	f[v]
P	27	28

The sort is the list of vertices in the reverse order of their finish. In the algorithm when a vertex finishes we place it at the *start* of a linked list, initially nil. At the end, with negligible extra time, we have the list:

$$PNOSMRYVXWZUQT$$

7. Let $S(n)$ satisfy initial condition $S(1) = 4$ and recursion $S(n) = S(n/7) + 11$ Assume n is a power of 7. Give a *precise* formula for $S(n)$.
`Solution:` We get from $S(1)$ to $S(n)$ in $\log_7 n$ steps – where a step is going from $S(x)$ to $S(7x)$. Each time we add 11 so we added a total of $11 \log_7 n$. We started with $S(1) = 4$. So $S(n) = 4 + 11 \log_7 n$.

8. **Not to be Submitted!** If one person is purple on December 10, 2020 and the number of purple people doubles every five days, at what day does the number of purple people reach $7 \cdot 10^9$?
`Solution:` Ten doublings makes a thousand, twenty a million, thirty a billion so 33 doublings reaches $7 \cdot 10^9$. That is 165 days. So $31 + 31 + 28 + 31 + 30 = 151$ days for five months, until May 10, 2021 and then another 14 days, a purple world on May 24, 2020.