CSE 373 Analysis of Algorithms April 23, 2015

Solutions to Midterm 2

Problem 1

(1) $A, B, E, C, F, I, D, G, J, M, H, K, N, L, O, P$
(2) $A, B, C, D, H, G, F, E, I, J, K, L, P, O, N, M$
(3) The edges we are using:

$$(A, B), (A, E), (B, F), (E, I), (F, J), (I, M), (M, N)$$

$$(J, K), (N, O), (C, G), (G, K), (D, H), (H, L), (L, P), (P, O)$$

(4) The edges we are using:

$$(A, E), (E, I), (I, M), (M, N), (N, O), (O, P), (P, L)$$

$$(A, B), (B, C), (C, D), (B, F), (F, G), (G, H), (F, J), (J, K)$$

Problem 2

Say we have m prime factors $[p_1, p_2, \cdots, p_m]$, and the exponents for these prime factors in n are $[e_1, e_2, \cdots, e_m]$ respectively. Then, the solution vector should be in the form of $[c_1, c_2, \cdots, c_m]$, in which each c_i denotes the exponent we use for prime factor p_i. Thus, the corresponding divisor for this solution vector is:

$$d = p_1^{c_1} p_2^{c_2} \cdots p_m^{c_m}$$

For each function:

(1)
```
/*here n is the number of prime factors*/
construct_candidates(int a[], int k, int n, int c[], int *ncandidates)
{
    for (int i = 0;i <= e[k];++i)
        c[i] = i;
    *ncandidates = 1 + e[k];
}
```

(2)
```
process_solution(int a[], int k)
{
    int product = 1;
    for (int i = 1;i <= m;++i)
        product *= pow(p[i], a[i]);
    print (product);
}
```

(3)
```
is_a_solution(int a[], int k, int n)
{
    return (k == n);
}
```

Problem 3

(1) For $n = 9$: the maximum product is $3 \cdot 3 \cdot 3 = 27$.
For $n = 12$: the maximum product is $3 \cdot 3 \cdot 3 \cdot 3 = 81$.

(2) Say $product(n)$ is the maximum product we can get for n. Then by enumerating the first cut we make, we have the following recurrence:
$product(n) = max(product(n-i) \cdot i), i \in [1, n-1]$
The base cases are: $product(n) = n, i \in [1, 3]$. Time complexity is $O(n^2)$.

Problem 4

(1) Assign red edge with weight of 1, green edge with weight of 0. Run Prim or Kruskal, the minimum spanning tree uses the minimum number of red edges.

(2) Assign red edge with weight of 1, green edge with weight of 0. Run Dijkstra's algorithm, the shortest path uses the minimum number of red edges.

Problem 5

Run topological sort on the DAG, which takes $O(n+m)$. Then check if there is an edge between each pair of adjacent vertices in the topological order. If it is true, then the DAG contains a Hamiltonian path; else, it doesn't contain a Hamiltonian path. In total it takes $O(n+m)$.