Computer Science 3711
Winter 2004

Final Exam

Answers

1. (16 marks)

a) (10 marks) Circle the letters associated with the appropriate answers in the following multiple choice questions. Note that some of these questions may have more than one answer whose letter needs to be circled.

i) (6 marks) Which of the following is true of the function $T(n) = 3^{n+c}$ where c is an integer constant greater than zero?

a) $T(n)$ is $\Theta(2^{2^n})$ b) $T(n)$ is $\Omega(c^n)$ c) $T(n)$ is $\Theta(9^{\frac{n}{2}})$

d) $T(n)$ is $\Omega(2^n)$ e) $T(n)$ is $O(c^{n+3})$ f) $T(n)$ is $O(n^c)$

Answer: Parts (c) and (d) are circled.

ii) (4 marks) Which of the following is true of the function $T(G) = |V|^2|E| \log_2 |E|^3$ relative to directed graphs $G = (V, E)$?

a) $T(G)$ is $O(|V|^2 \log_2 |V|^5)$

b) $T(G)$ is $\Omega(|V|^2 \log_2 |V|^5)$ for directed complete graphs

c) $T(G)$ is $O(|V|^2 \log_2 |V|^2)$ for directed acyclic graphs

d) $T(G)$ is $\Omega(|E|^2 \log_2 |E|)$ for directed acyclic graphs

Answer: Parts (b) and (d) are circled.

b) (6 marks) Give a recurrence for the worst-case time complexity of the following recursive algorithm:

```
procedure FUNKY-REC(m, n)
    if (n < 1)
        return(m)
    else if (n > 4)
        mult = 10
        for i = 1 to m * m do
            l = FUNKY-REC(m, n/4)
            mult = FUNKY-ITR(i * m) * l
        return(mult)
    else
        mult = 1
        for (i = 1; i <= n * n * n; i++)
            for (j = 1; j <= m; j++)
                mult = mult * FUNKY-ITR(i)
        return(mult)
```

Assume that procedure `FUNKY-ITR(x)` runs in $O(x^2)$ time.

Answer:

$$T(m,n) = \begin{cases} O(1) & n < 1 \\ O(m) & 1 \leq n < 4 \\ m^2 T(m, \frac{n}{4}) + O(m^8) & n \geq 4 \end{cases}$$

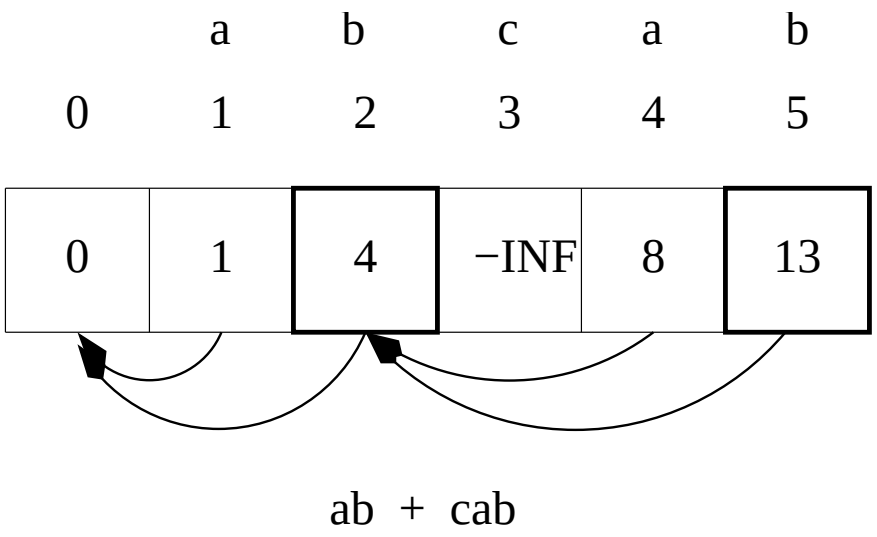
2. (24 marks)

a) (12 marks) Consider the variant on the dynamic programming algorithm for Assignment #3 as defined by the recurrence

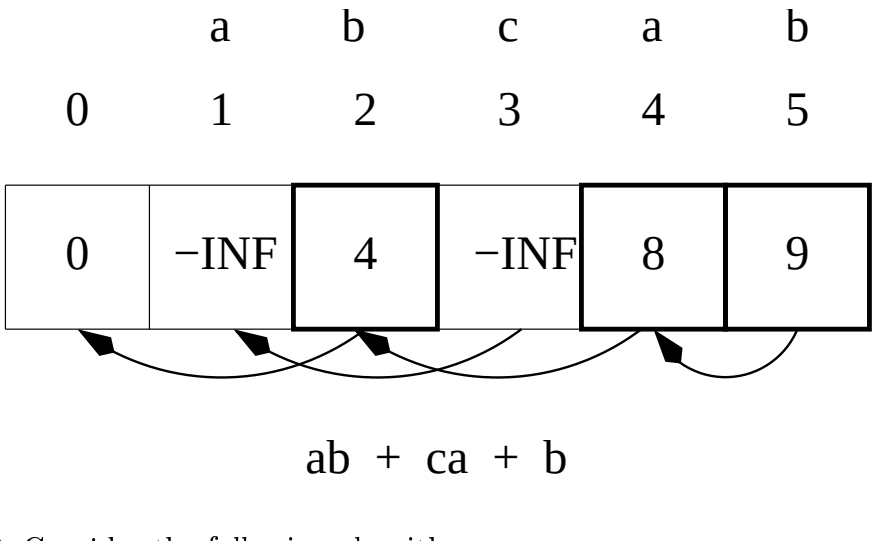
$$C[i] = \begin{cases} 0 & i = 0 \\ \max_{d \in D \text{ and } d \in suff(i)} C[i - |d|] + |d|^2 & \text{if } i > 0 \text{ and } suff(i) \neq \emptyset \\ -\infty & \text{otherwise} \end{cases}$$

where D is the codeword dictionary, $c(i)$ is the string composed of the first i characters of the ciphertext c, and $suff(i)$ is the set of suffixes of $c(i)$. This variant essentially searches for the decryption of c relative to D with the fewest number of large codewords. Given the ciphertext $c = abcab$, compute and output the optimal decryption of c relative to the recurrence above (that is, fill in the given dynamic programming table (including one backpointer per table-cell), show the backpointer path that gives an optimal decryption, and give the decryption associated with that backpointer-path) for the following codeword dictionaries:

i) (6 marks) $D = \{a, b, ab, ba, ca, cab\}$



ii) (6 marks) $D = \{b, ab, bc, ca, bca\}$



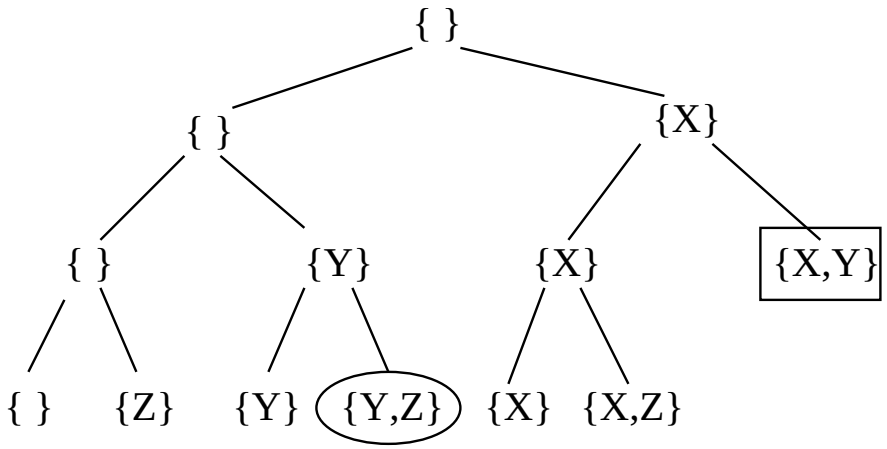
b) (12 marks) Consider the following algorithm:

```
DFS-V(i, sol, U, B, val)
    if (SIZE(sol) > B)
        print("pruned: solution too big for knapsack")
    else if (i == n)
        if (VALUE(sol) >= val)
            print(sol)
    else
        DFS-V(i + 1, sol, U, B, val)
        DFS-V(i + 1, UNION-COPY(sol, U[i]), U, B, val)
```

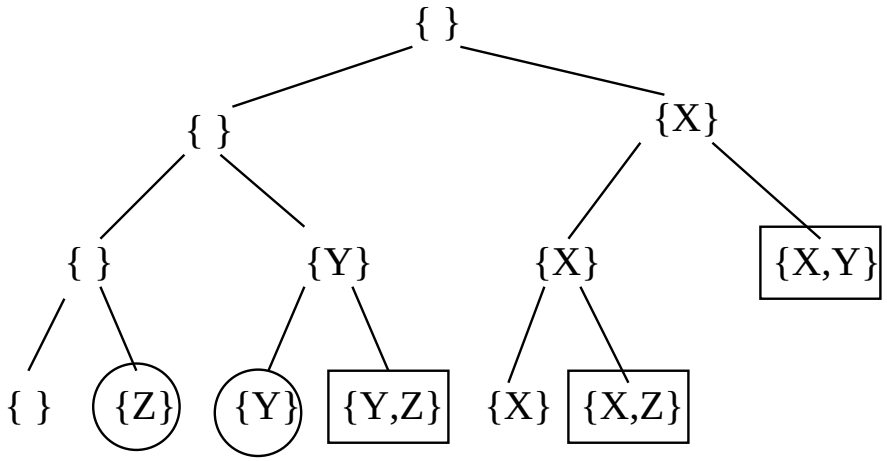
The problem solved here is a variant of 0/1 KNAPSACK that prints all viable

knapsack-loads of value at least val. In this algorithm, U is the set of items, n is the number of items in U, B is the knapsack size bound, sol is a subset of U, SIZE(sol) returns the sum of the sizes of the items in sol, VALUE(sol) returns the sum of the values of the items in sol, and UNION-COPY($sol, U[i]$) returns a copy of sol to which the ith item in U has been added. To answer this question, sketch the implicit tree of solution-nodes generated by the algorithm above (marking leaf-nodes which are printed by a circle and nodes that are pruned with a square) with the call DFS-V($0, sol, U, B, val$) when $U = \{X, Y, Z\}$ such that $size(X) = 2$, $size(Y) = 2$, $size(Z) = 1$, $value(X) = 1$, $value(Y) = 3$, and $value(Z) = 2$, sol is the empty set, and B and val have the following values:

i) (6 marks) $B = 3$ and $val = 4$:

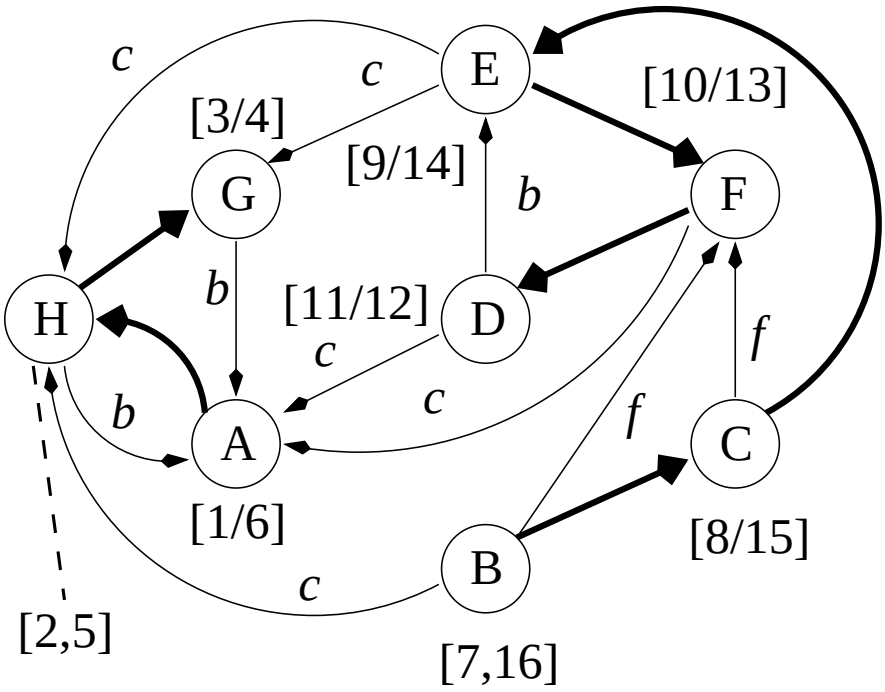


ii) (6 marks) $B = 2$ and $val = 2$:



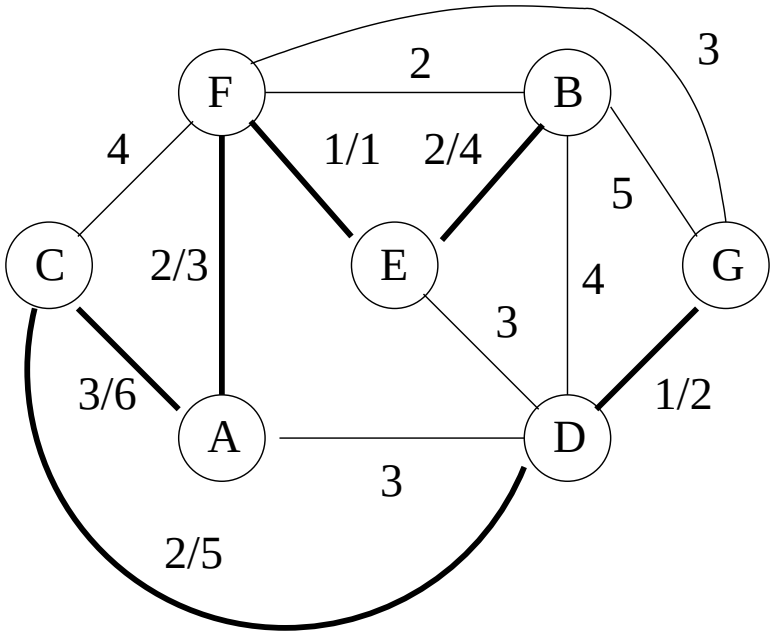
3. (24 marks)

a) (8 marks) Consider the following directed graph:

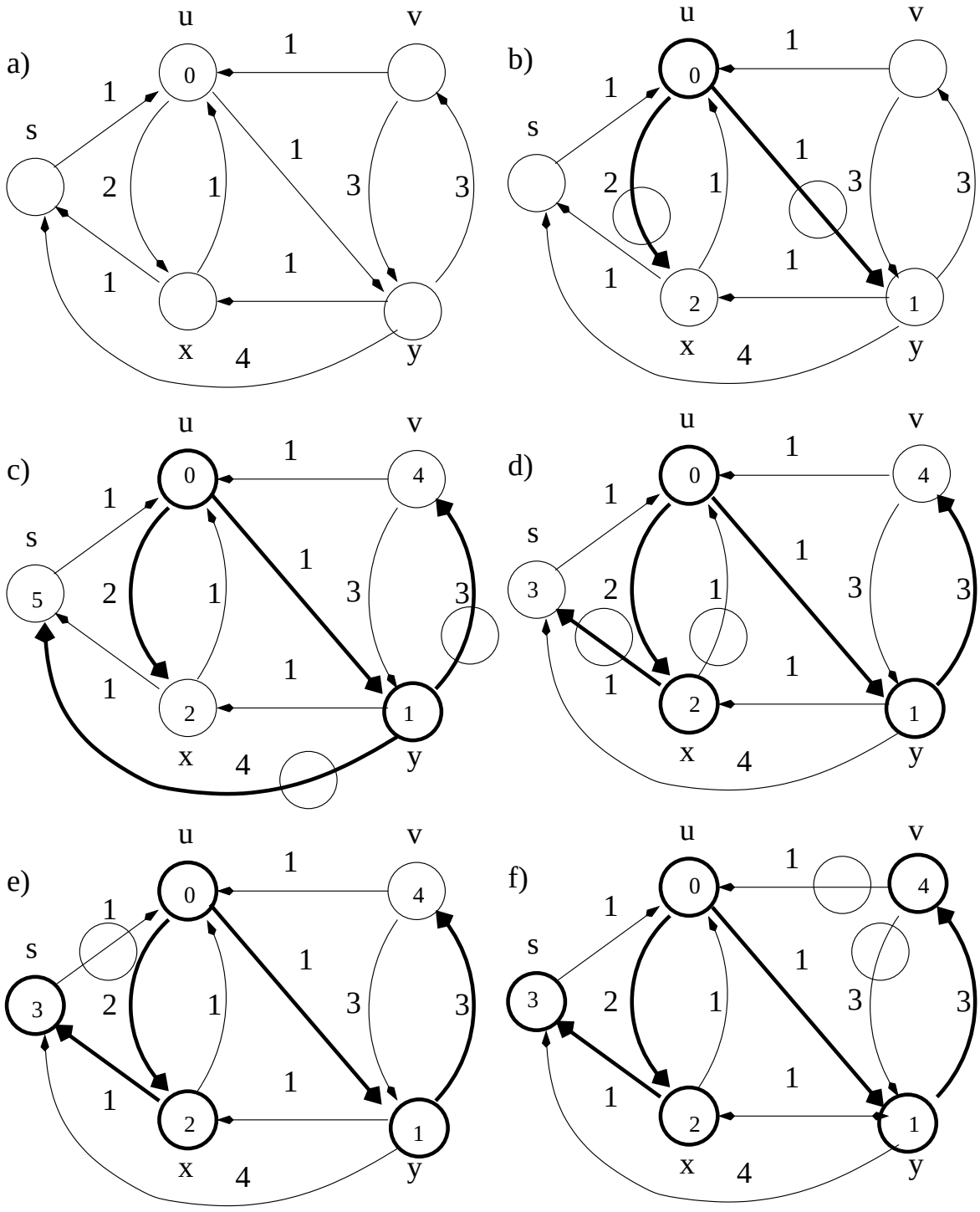


Give the graph at the end of the execution of the DFS algorithm, with the d- and f-values of all vertices as well as the types of all edges clearly marked. Assume that the algorithm considers vertices in alphabetical order and that each adjacency list is ordered alphabetically.

b) (6 marks) Show how Kruskal's minimum spanning tree algorithm works on the undirected edge-weighted graph below. Mark all tree-edges and the order in which each tree-edge was added.



c) (10 marks) Run Dijkstra's algorithm on the directed graph below using vertex u as the source vertex. Show the d and π values and the vertices in set S after each iteration of the **while** loop by filling in the appropriate values on the spare copies of the graph given in this table.



4. (36 marks)

a) (18 marks) Consider the following problem: For the last two weeks, your favorite nephew has been mansion-sitting while you have been in the Far East exploring new feeding options for your guest house. On your return late at night, you find your nephew along with a collection of his semi-comatose "friends" passed out in your foyer amidst the remains of the contents of your wine cellar. After waking your nephew in a somewhat less than gentle fashion, you and he decide to bring as many of his friends as possible back to their respective homes in your Jaguar, two at a time. Each of the friends lives in a distinct place. Your Jaguar's gas tank is full (thank goodness) and you know the distance from your mansion to each friend's home as well as the distance between each pair of homes.

Give pseudocode for a polynomial-time algorithm that solves the problem above – namely, given an edge-weighted undirected graph $G = (V, E, w)$ representing the road-map, a vertex $M \in V$ representing the location of your mansion, a set of vertices $F \subset V - M$ representing the locations of the friends' homes, and an integer $n > 0$ representing the distance your Jaguar can cover before it runs out of gas, find the largest set of pairs $T = \{(f_{11}, f_{12}), (f_{21}, f_{22}), \ldots, (f_{k1}, f_{k2})\}$ such that $f_{ij} \in F$ for $1 \leq i \leq k$ and $j \in \{1, 2\}$ and $\sum_{i=1}^{k}(sp(M, f_{i1}) + sp(f_{i1}, f_{i2}) + sp(f_{i2}, M)) \leq n$, where $sp(x, y)$ is the shortest (in terms of summed edge-weight) simple path between x and y in G. Please give the asymptotic worst-case time complexity of your algorithm.

Answer: This problem is not solvable in polynomial time, and hence it was an error on my part to ask for such an algorithm, *i.e.*, everyone who wrote this exam got 18 "free" marks. Treat this as a cautionary tale of how careful you have to be when you are defining problems.

b) (18 marks) Let $G = (V, E, w, l, U)$ be an edge-weighted directed graph such that each edge $e \in E$ has weight $w(e)$ and each vertex v has an associated subset $l(v)$ of a set of items U. Given an item $u \in U$, you want the set of ordered vertex-triples (v_1, v_2, v_3) such that:

1. $u \in l(v_1)$,
2. the edges (v_1, v_2) and $v_2, v_3)$ are in E, and
3. the value $w((v_1, v_2)) * w((v_2, v_3))$ is the largest possible over the set of all ordered vertex-triples in G that satisfy (1) and (2) above.

Give pseudocode for a polynomial-time algorithm that solves this problem. Please give the asymptotic worst-case time complexity of your algorithm.

Answer: The key here is to first determine the set of all vertex-triples that satisfy conditions (1) and (2) above and then to find those triples in this set that have the maximum value. This can be done by the following algorithm:

```
Create empty list A
for each vertex v1 in V do
    if l is in l(v1) then
        for each vertex v2 adjacent to v1 do
            for each vertex v3 <> v1 adjacent to v2 do
                Add the four-tuple (v1, v2, v3 val) to A
                 where val = w((v1, v2)) * w((v2, v3))
Sort A in descending order by the fourth value of each tuple
Set max to the fourth value of A(1)
print A(1)
i = 2
while the fourth value of tuple A(i) = max do
    print A(i)
```

The first group of loops run in $O(|V|^3 L)$ time, where L is the maximum number of labels associated with any vertex in V. As there are at most $|V|^3$ tuples in A at the end of the first group of nested loops, A can be sorted in $O(|V|^3 \log_2 |V|^3) = O(|V|^3 \log_2 |V|)$ time and the final `while`-loop runs in $O(|V|^3)$ time. Hence, the algorithm as a whole runs in $O(|V|^3 L + |V|^3 \log_2 |V|) = O(|V|^3 (L + \log_2 |V|))$ time.

5. (20 marks) Circle the letters associated with the appropriate answers in the the multiple choice questions in parts (i–v). Note that some of these questions may have more than one answer whose letter needs to be circled.

Suppose we have seven decision problems A, B, C, D, E, F, and G such that $A \leq_p C$, $A \leq_e F$, $B \leq_p A$, $C \leq_p E$, $C \leq_p G$, $D \leq_e A$, $E \leq_p F$, and $F \leq_e E$, where $X \leq_p Y$ ($X \leq_e Y$) means that there is a polynomial-time (exponential-time) many-one reduction from X to Y, *i.e.*, X reduces to Y.

i) (4 marks) What is implied if we know that A is solvable in polynomial time?

a) B is not solvable in polynomial time.
b) if B is solvable in polynomial time then $P = NP$.
c) D is solvable in exponential time.
d) F is solvable in exponential time.

ii) (4 marks) What is implied if we know that A is NP-complete?

a) G is NP-hard.
b) E is NP-complete.
c) F is NP-hard.
d) B is solvable in polynomial time.

iii) (4 marks) What is implied if we know that C is NP-hard and F is solvable in polynomial time?

a) D may be solvable in polynomial time.
b) $P = NP$.
c) B is solvable in polynomial time.
d) E is NP-complete.

iv) (4 marks) What is implied if we know that F is NP-hard and E is solvable in polynomial time?

a) G is solvable in polynomial time.
b) $P \neq NP$.
c) A is solvable in exponential time.
d) F may be solvable in polynomial time.

v) (4 marks) What is implied if we know that B is NP-hard, G is solvable in polynomial time, and E is in class NP?

a) F is solvable in polynomial time.
b) $P = NP$.
c) D is solvable in exponential time.
d) C may be solvable in polynomial time.

Answer: In this question, polynomial-time tractability (intractability) results are propagating backwards (forwards) along chains of polynomial-time many-one reductions. The easiest way to untangle what is going on is to draw a directed graph of the given reductions, where the vertices of this graph are problems and there is an arc from X to Y if there is a reduction from problem X to problem Y. Given such a graph, we can then label the vertices with the appropriate results and propagate these results accordingly.

The reduction-graph corresponding to the situation described in this question is given in Part (a) of Figure 1. Given the results described in parts (i–v), we can then deduce that the following implications are true:

Part (i): (c) (see part (b) of Figure 1)

Part (ii): (a) and (c) (see part (c) of Figure 1)

Part (iii): All (see part (d) of Figure 1)

Part (iv): (c) and (d) (see part (e) of Figure 1)

Part (iv): (b), (c) and (d) (see part (f) of Figure 1)

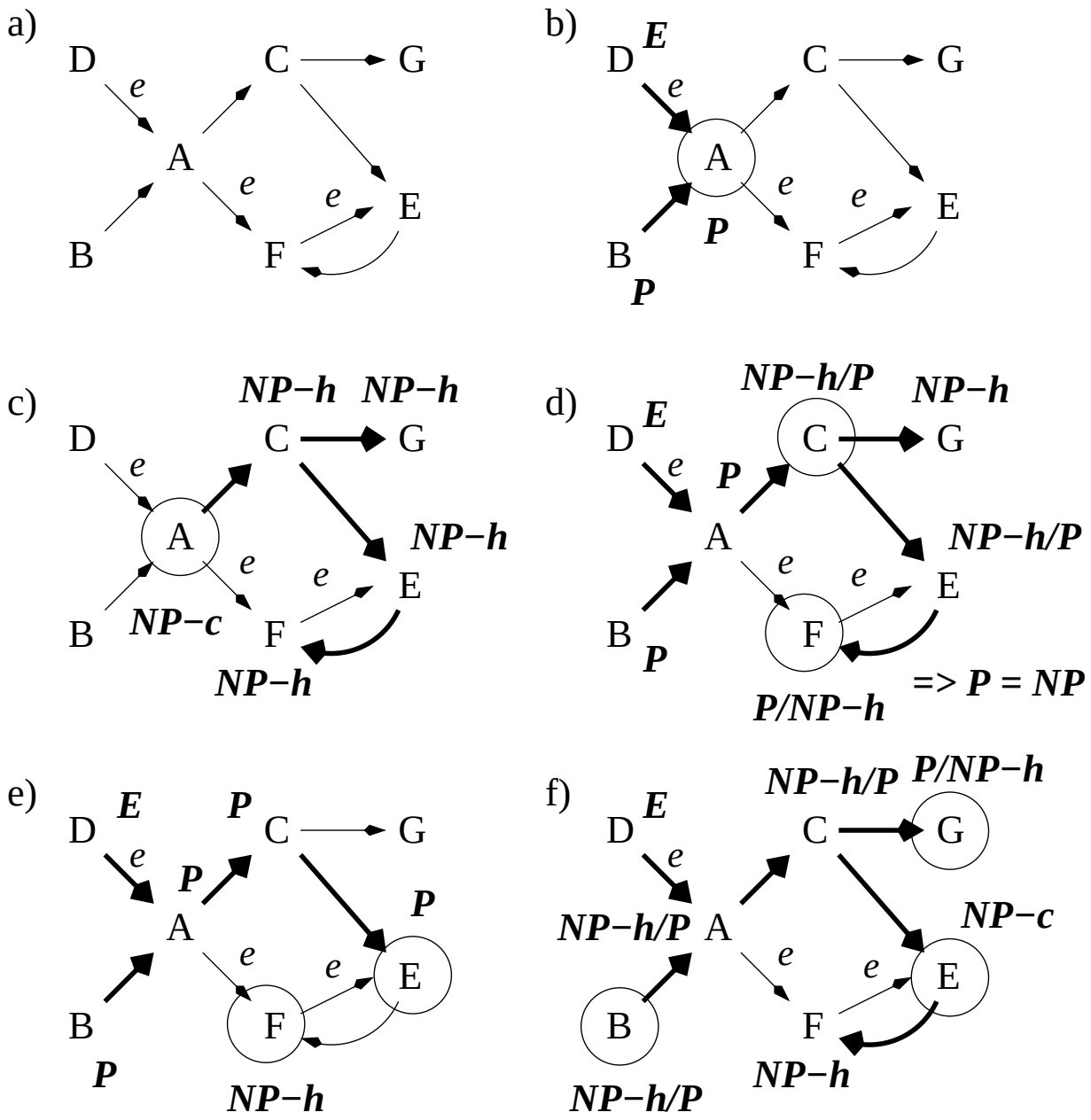


Figure 1: Answers for Question #5. **(a)** Reduction-graph for question. **(b – f)** Reduction-graphs relative to results described in subparts (i – v). Reductions along which results propagate are highlighted.