CS 692 Capstone Exam Algorithms Fall 2020: Choose any 2 of the 3 problems.

1) A "full node" in a binary tree has either no children or exactly 2 children. Given a binary tree, write a function that returns true if all nodes are full nodes. Otherwise return false.

Notes: The function should have just one argument, a pointer to the root.
No global variables may be used.
No additional functions may be defined.

2) For each function with input argument n, determine the asymptotic number of "fundamental operations" that will be executed. Note that `fc` is recursive. Choose each answer from among the following. You do not need to explain your choices.

$\theta(1)$ $\theta(\log n)$ $\theta(n)$ $\theta(n \log n)$ $\theta(n^2)$ $\theta(n^2 \log n)$ $\theta(n^3)$ $\theta(2^n)$ $\theta(n!)$

a)
```
void fa(int n) {
  for(i = 1; i < n; i++) {
    for(j = i; j <= n; j++)
      Perform 1 fundamental operation;
      //endfor j
  }//endfor i
  for(k = 1; k <= n; k++)
    Perform 1 fundamental operation;
  //endfor k
}
```
b)
```
void fb(int n) {
  for(i = 1; i <= n; i = 3*i)
    Perform 1 fundamental operation;
  //endfor i
}
```
c)
```
void fc(int n) {
  if (n > 1) {
    fc(n/2);
    fc(n/2);
    Perform 1 fundamental operation;
  }//endif
}
```
d) For part d), a fundamental operation will be a comparison of array element types.
```
void hybridsort(element a[], int n){
// Put a[0]..a[n-1] into ascending order.
  if (n < 100)
    Perform selection sort;
  else
    Perform mergesort;
  //endif
}
```

3) Solve the recurrence relation $T(n) = T(n/2) + 3n$ where $T(1) = 0$ and $n = 2^k$ for a nonnegative integer k. Your answer should be a precise function of n in closed form (i.e., resolve all sigmas and …'s). An asymptotic answer is not acceptable. Justify your solution.

CS 692: Theory Exam
Fall 2020

Answer any **TWO** of the three problems listed below. If you attempt all three, only questions one and two will be graded. Please show all work.

Q1. Answer the following questions. In all cases Σ = {a, b}. **(10 points each).**

a) Let G be the grammar below:

$$S \rightarrow aS \mid aB$$
$$B \rightarrow bB \mid b$$

Let L(G) be the language generated by grammar G.
Either prove that L(G) is regular by providing a regular expression that describes L(G) or disprove it by applying the pumping lemma.

b) Let G' be the grammar below:

$$S \rightarrow aSb \mid ab$$

Let L(G') be the language generated by grammar G'.
Either prove that L(G') is regular by providing a regular expression that describes L(G') or disprove it by applying the pumping lemma.

Note: If you are using the pumping lemma to disprove the regularity of anyone of the above cases, you first need to clearly define the lemma. Then use it to disprove.

Q2. Provide a **context-free grammar** for each of the following languages. In all cases Σ = {0,1}. **(each 10 points)**

a. L= {w| w = w^R and |w| is even}, where **w = w^R** means w is palindrome (reads the same forward and backward)

b. L= {w| w starts and ends with the same symbol}

Q3) Answer the following questions. Please clearly explain each in detail. (10 points each)

a) Consider the Vertex Cover (VC) problem defined as follows:
A vertex-cover of an **undirected** graph G = (V, E) is a subset of vertices **V' ⊆ V** such that if edge (u, v) is an edge in E, then either **u** ∈ **V'** or **v** ∈ **V'** or both.

VC = {G, k : G = (V, E) is an undirected graph containing a vertex-cover of size k}.

Prove that Vertex Cover (VC) is in NP.

Please note: You are being asked to prove that VC is in NP. You do NOT need to prove that VC is NP-Complete.

b) How do you prove, in general, that a Problem X is NP-complete? Please give the steps and explain.

1) **Virtual Memory – 20pts total**

a. (4pts) Assuming a virtual memory system with FIFO page replacement and an arbitrary page access pattern, will increasing the number of page frames decrease the number of page faults? Why or why not?

b. (4pts) Assume a 32 bit logical address space and three level paging system. The first 12 bits are for the 1st level page table, the next 8 bits are for the 2nd level page table, the next 6 bits are for the 3rd level page table and remaining 6 are for the offset. How much virtual memory can be accessed?

c. (4pts) Assume a 32 bit virtual memory system with a page size of 16KB. The Translation Lookaside Buffer (TLB) can hold 512 page table entries . What is the minimum size of the TLB tag?

d. (4pts) Given a 4 level page table with a Translation Lookaside Buffer (TLB) hit ratio of 92%, What is the effective access time given that a TLB access is 80ns and a memory access time is 160ns?

e. (4pts) If a machine has 64MB of physical memory and a 32 bit virtual address space and a page size of 8KB, what is the size of the page table? You must show your calculations for full credit.

2. **Resource Allocation (20 pts Total)**

a. (4pts) List the four standard necessary conditions of Deadlock.

b. (3pts) Releasing all resources before requesting a new resource is a valid deadlock prevention scheme. Is this statement True or False? Explain your answer.

c. (3pts) In a Resource Allocation Graph (RAG), where there is one instance of each resource, a cycle implies that there is deadlock. Is this statement true or false? Explain your answer.

d. (4pts) A system with 5 threads contains 5 instances of resources. Let T be a set of threads and R be a set of resources. R1 has 2 instances. R2 has 1 instance, and R3 has 2 instances

The sets P and R are as follows:

T = {T1, T2, T3, T4, T5}
R = {R1, R2, R3, R4, R5}

.

T1 is assigned an instance of R3 and wants an instance of R1
T2 is assigned an instance of R1
T3 is assigned an instance of R1 and wants an instance of R2
T4 is assigned an instance of R2 and wants an instance of R3
T5 is assigned an instance of R3

Show the set E based on the above assignments and wants.

(6pts) Is there deadlock in this situation? Briefly explain your answer.

3. Mixed (20pts total)

a. (4pts) Explain the difference between a modular kernel design and a monolithic one. What is an advantage of modular design?

b. (3pts) Never requesting a resource after releasing a resource is a valid deadlock prevention scheme. Is this statement true or false? Explain your answer.

c. (4pts) In order to guarantee mutual exclusion, what should the **condition** of the while loop be in the code below?

```
repeat
  flag[ i ] = true;
  turn = J;
  while (  condition ) do;
       //critical section
       flag[ i ] = false;
       //non critical section
until false;
```

d. Given the code below: Assume the common variable lock is initially false and assuming testSet is an indivisible function that returns the value of its Boolean argument and sets the argument to True.

```
Process A
while (True)
{
   while (testSet(lock));
 // critical section
    lock = false;
    // Non critical section
}
```

```
Process B
while (True)
{
while (testSet(lock));
      // critical section
      lock = false;
      // Non critical section
}
```

(3pts) Does the code guarantee mutual exclusion? Explain your answer.

(3pts) Is it possible processes will busy wait forever? Explain your answer

(3pts) Is indefinite postponement possible? Explain your answer.