# 國立陽明交通大學 試 題 紙

一一二學年度第二學期
博士班資格考

科目：作業系統 A　　　　日期： 113 年 7 月 3 日　第 1 頁 共 1 頁

請 "✓" 明　　✓不可看書　　可看書

* 請將答案依題號順序寫入答案卷

答題時字跡需工整，否則不予計分。Write your answers legibly; otherwise you will get zero score.

1. (6%) What are the internal and external fragmentations?

2. (6%) What are the three typical types of i/o devices? Give an example of each device.

3. (2%) The File Allocation Table of Microsoft's FAT32 file system is a variation of:
   (A) contiguous allocation.
   (B) indexed allocation.
   (C) linked allocation.
   (D) combined indexing.

4. (5%) What is the race condition?

5. (5%) Given a UNIX i-node with ten direct blocks, one single indirect block, one double indirect block and one triple indirect block. Assume that the size of a block and the block address are 16Kbytes and 4 bytes, respectively. What would be the size of the largest file allowed in byte?

6. (6%) Briefly describe the concept of "thread", "process" and "program".

7. (6%) What is the problem with priority scheduling? What is the solution?

8. (9%) Given a disk with 200 tracks, numbered from 0 to 199, Assume the disk head is initially located at track 120 and was moving in the direction of increasing track number. Let the requested tracks, in the order received by the disk scheduler, be 55,58, 39, 18, 90, 160, 150, 38, and 186. What is the order that the requests are serviced by using the SCAN, C-SCAN, and LOOK scheduling algorithms, respectively?

9. (5%) How many processes does the following code segment create including the original process?

```
void main() {
   for ( int i = 0 ; i < 3 ; i++ ) {
      if ( fork() == 0 ) {
         fork() ;
      } // if
   } // for
} // main()
```

# 國立陽明交通大學 試 題 紙

一一二學年度第二學期
博士班資格考

科目：作業系統 B　　　　日期： 113 年 7 月 3 日　第 1 頁 共 1 頁

請"✓"明　　✓不可看書　　可看書

* 請將答案依題號順序寫入答案卷

答題時字跡需工整，否則不予計分。Write your answers legibly; otherwise you will get zero score.

1. [3pts] When a user process runs a system call, please explain how the user process notifies the OS kernel.
2. [4pts] Please describe how the OS kernel handles the system call.
3. [5pts] Please briefly describe the interrupt procedure between a hardware device and the CPU.
4. [5pts] Linux creates a new process using the fork() and exec() system calls. Please briefly describe the functionalities of fork() and exec().
5. [3pts] Please describe the purpose of virtual memory.
6. [5pts] Please briefly describe when the CPU triggers a minor page fault and when it triggers a major page fault.
7. [5pts] Modern x86 systems usually adopt multi-level page tables. How many minimum levels are there for the system where the page size, virtual address space size, and page table entry size are set to 4KB, 258B, and 16B, respectively?
   (Please show the calculation process. No points without the calculation process.)

8. [20pts] Please answer **True** or **False** for each statement below:
   a. Threads belonging to the same process share the code section, data section, stack, heap and file descriptors.
   b. When an x86 CPU encounters a logical address that results in a TLB miss, the CPU doesn't require the execution of Linux to walk the page table for translating to a physical address.
   c. After activating virtual memory, Linux kernel merges kernel and user address space together so that the user process can receive a larger memory space.
   d. With the support of virtual memory, different user processes can store different data at the same virtual address within their own address space.
   e. In the same file system, two files in different directories can have the same name but they shall have different inode numbers.
   f. It's feasible for two different processes to get identical file descriptor numbers, and yet these specific descriptors may reference the same file.
   g. When the data requested by a CPU is not in the DRAM, DRAM interrupts CPU. CPU preserves the state of the user code and directs the execution to the kernel routine or thread responsible for fulfilling the page fault handling service.
   h. Device controllers run the device driver to interact with the Linux kernel. Thus, programmers do not need to understand the details of how the device works.
   i. On the operating system adopting paging (e.g., Linux), multiple processes can share one page table to save memory consumption.
   j. To achieve better performance on virtual memory systems, programmers manage page frames by considering process behaviors, such as spatial and temporal access locality.

# 國立陽明交通大學 試 題 紙

一一二學年度第二學期
博士班資格考

科目：計算理論 A　　　　日期： 113 年 7 月 2 日　第 1 頁 共 1 頁

請 "✓" 明　　✓不可看書　　可看書

* 請將答案依題號順序寫入答案卷

答題時字跡需工整，否則不予計分。Write your answers legibly; otherwise you will get zero score.

1. (15%) Write a deterministic Turing machine in state diagram to accept the language $L$ of balanced parentheses. For example, $(()())()$ and $((())())$ are in $L$, while $())()$ and $(()()))$ are not.

2. (15%) Show that the set $L = \{\langle M_1, M_2\rangle : L(M_1)$ is recursive or $L(M_2)$ is not recursive $\}$ is not recursively enumerable, where $\langle\cdot\rangle$ is the standard encoding of Turing machines.

3. (20%) This problem is to prove that the vertex cover problem $VC$ is $NP$-complete, where $VC = \{\langle G, k\rangle : G = (V, E)$ is an undirected graph that has a $k$-node vertext cover $\}$.

   (a) Show that $VC$ is in NP.

   (b) Give a poly-time reduction $\phi$ from $3SAT$ to $VC$, where $3SAT$ is a satisfiable Boolean formula $F = \bigwedge_{i=1}^{m}(l_{i,1} \vee l_{i,2} \vee l_{i,3})$ in conjunctive normal form over $n$ variables $x_1, x_2, \ldots, x_n$. That is, for given $\langle X, F\rangle$, compute $\phi(\langle X, F\rangle) = \langle G, k\rangle$ in polynomial time.

   (c) Show that there is a truth assignment for $X$ that satisfies $F$ if and only if $G$ has a $k$-node vertex cover.

# 國立陽明交通大學 試 題 紙

一一二學年度第二學期
博士班資格考

科目：計算理論 B　　　　日期：113 年 7 月 2 日　第 1 頁 共 1 頁

請 "✓" 明　　✓不可看書　　可看書

* 請將答案依題號順序寫入答案卷

答題時字跡需工整，否則不予計分。Write your answers legibly; otherwise you will get zero score.

1. (10%) Prove or disprove that if $L$ is a regular language, the complement of $L$ is also regular.
2. (10%) Prove or disprove that if $L$ is a context-free language, the complement of $L$ is also context-free.
3. (10%) Prove or disprove that $L = \{ x \mid x = ww, w \in \{0, 1\}^* \}$ is regular.
4. (10%) Prove or disprove that $L = \{ x \mid x = ww, w \in \{0, 1\}^* \}$ is context-free.
5. (10%) Prove or disprove that $L = \{ x \mid x \neq ww, w \in \{0, 1\}^* \}$ is context-free.

# 國立陽明交通大學 試 題 紙

一一二學年度第二學期
博士班資格考

科目：計算機架構 A　　日期：113 年 7 月 2 日　

請 "✓" 明　　✓不可看書　　可看書

* 請將答案依題號順序寫入答案卷

答題時字跡需工整，否則不予計分。Write your answers legibly; otherwise you will get zero score.

1. (8%)
   (a) (3%) Describe the three main concerns from the system perspective of power and energy.
   (b) (2%) Describe ***Amdahl's Law*** and define the equation of the overall speedup using the original machine with the enhanced mode.
   (c) (3%) Define the following two main measures of dependability: ***module reliability*** and ***module availability***. Write the equations with respect to *mean time to failure* (*MTTF*) and *mean time to repair* (*MTTR*) for measuring these two quantities of a non-redundant system with repair.

2. (6%) A 2-GHz processor was used to execute a benchmark program with the instruction mix and clock cycle counts shown in the following table.

| Instruction type | Frequency | CPI |
|---|---|---|
| Integer ALU ops | 20% | 1 |
| Floating-point ops | 40% | 20 |
| Loads/Stores | 30% | 4 |
| Branches | 10% | 2 |

   (a) (2%) Assume that the total number of instructions executed is $4\times10^9$, determine the ***effective CPI*** (cycles per instruction), and ***execution time*** of this program.
   (b) (2%) Assume that an enhancement proposal is to reduce the CPI of the floating-point operations to 8 with 20% lengthening of the clock cycle time. Calculate the ***effective CPI*** and the ***speedup*** of the enhancement.
   (c) (2%) Calculate the ***fraction of the time*** for executing load and store operations of this program in the original processor. Assume that we build an optimizing compiler to discard half (1/2) of the Load/Store operations from the original instruction mix, determine the ***speedup*** of the enhancement to the original design by applying **Amdahl's Law**.

3. (8%)
   (a) (2%) Describe the main characteristic of program access, including instructions and data, that make ***memory hierarchy*** design feasible and efficient.
   (b) (2%) Explain the main reason that make the strategies adopted for the design of ***cache*** and ***virtual memory*** different,
   (c) (4%) Describe the policies adopted for ***block placement***, ***block identification***, ***block replacement***, and ***write strategy*** of a **paged virtual memory**.

# 國立陽明交通大學 試 題 紙

一一二學年度第二學期
博士班資格考

科目:計算機架構 A　　日期: 113 年 7 月 2 日　

4. (7%)

(a) (3%) Assume that the CPU address is N bits and byte addressable, the cache has $2^i$ blocks, the block size is $2^j$ bytes, and the cache has $2^k$ ways if it is set-associative. An exemplar address division diagram is given below. Determine the number of bits of the tag, index, and block offset fields for each of the following block placement approaches: ***direct mapped***, ***set associative***, and ***fully associative***.

| Block address | | Block offset |
|---|---|---|
| Tag | Index | |

(b) (4%) For a computer system, the processor is in-order execution that runs at 2.5 GHz and has a CPI (cycles per instruction) of 1.4 excluding memory stalls. Suppose that in 1000 memory references there are 80 misses in the first-level (L1) cache and 20 misses in the second-level (L2) cache. Assume that the hit time of L1 cache is 1 clock cycle, the hit time of the L2 cache is 20 *ns*, the miss-penalty from the L2 cache to memory is 400 *ns*, and there are 1.1 memory references per instruction. Assume that the miss penalty need not be rounded to an integral number of clock cycles. Ignore the impact of writes. Calculate the ***average memory access time*** in *ns* and the **overall CPI**, including memory stalls. Calculate the ***average memory access time*** in *ns* and the **overall CPI**, including memory stalls.

5. (10%) Typically, **miss rate**, **miss penalty**, **hit time**, and **bandwidth** are the main metrics to evaluate the performance of a cache memory. Describe each of the following cache optimizations and indicate its positive impact to the performance metrics:

(a) ***multilevel cache***

(b) ***nonblocking cache***

(c) ***virtually addressed cache***

(d) ***blocking*** (a compiler optimization)

(e) ***compiler-controlled prefetching***

6. (11%)

(a) (3%) Describe the following three variations of **SIMD**: ***vector architectures***, ***multimedia SIMD instruction set extensions***, and ***graphics processing units*** (***GPUs***).

(b) (8%) Describe each of the following optimizations of **vector architecture** and the problem it solves:

i. ***vector-length register***　　ii. ***predicate register***　　iii. ***memory bank***　　iv. ***stride***

# 國立陽明交通大學 試 題 紙

一一二學年度第二學期
博士班資格考

科目：計算機架構 B　　　　日期：113 年 7 月 2 日　

請 "✓" 明　　✓不可看書　　可看書

* 請將答案依題號順序寫入答案卷，第 2 到第 5 題請直接在試題紙上作答。

答題時字跡需工整，否則不予計分。Write your answers legibly; otherwise you will get zero score.

1. Fill-in-the-blank or circle the best answers. [10 points]

   i. In the MESI protocol, which of the following transitions might happen due to a transaction by another processor?

      M→E　　S→E　　I→S　　M→S　　S→M

   ii. Assume you have a pipelined multiplier on the critical path of your processor. If you increase the number of pipeline stages, what would you expect to happen?

      - The clock period and the CPI would go up.
      - The clock period would go up and the CPI would go down.
      - The clock period would go down and the CPI would go up.
      - The clock period and the CPI would go down.

   iii. Adding more reorder buffer entries will reduce the amount of time that the processor stalls due to **structural hazards / data hazards / control hazards** but may decrease the number of **branch delay slots / clock period / clock frequency**.

   iv. Which hardware structure in out-of-order processors ensures that instructions commit in order?

      - Scoreboard
      - Reorder buffer
      - Physical register file
      - Issue queue

   v. Consider a Simultaneous Multithreading (SMT) machine with limited hardware resources. Circle the hardware constraints that can limit the total number of threads that the machine can support. Briefly describe the reason.

      - Number of functional units
      - Number of physical registers
      - Data cache size
      - Data cache associativity

2. Branch prediction [10 points] **(Please write your answers directly on the exam paper)**

Consider a processor with a 1024-entry Branch History Table (BHT), indexed by the lower-order bits of the program counter (PC). The BHT implements a two-bit predictor with the state transition diagram shown below. The edges in the state transition diagram represent the branch outcomes that cause a transition. The predictor starts in the "00" state. The predictor predicts "taken" when it is in states "01", "10", and "11", and it predicts "not taken" when it is in state "00".

The processor is a simple 5-stage pipeline with no branch delay slots. Assume that at address 0x1000, there is a word-sized array of integers with the following data: [0x0, 0x0, 0x0, 0x1, 0x1, 0x1, 0x0, 0x1].

**Code sequence**

```
p_4:
ADDI R7, R0, 7
ADDI R11, 0x1000
loop:
SUBI R7, R7, 1
ANDI R8, R7, 1
b_0:
BEQZ R8, l_1
LW R12, 0(R11)
b_1:
BEQZ R12, l_2
l_1:
ADD R9, R9, R16
l_2:
ADDI R11, R11, 4
b_3:
BNEZ R7, loop
```

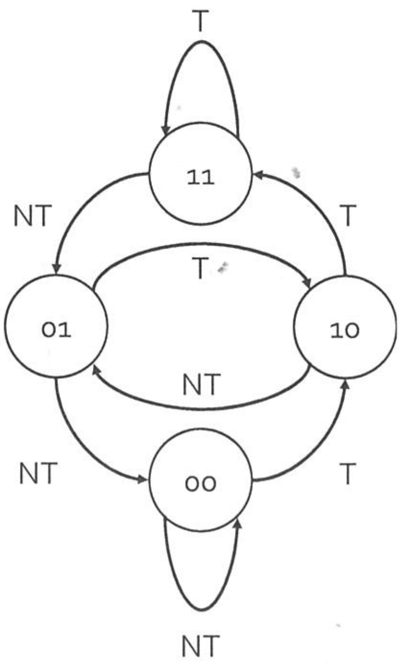


**Figure 1. Figure showing state machine**

Fill in the following table by simulating the execution of the above loop. Use 'x' to indicate an unknown value (e.g., if the branch is skipped).

| Machine State | | | | | BHT State | Branch Behavior | | Updated Values |
|---|---|---|---|---|---|---|---|---|
| PC | R7 | R8 | R11 | R12 | BHT Bits | Predicted Outcome | Actual Outcome | New BHT Bits |
| b_0 | 6 | 0 | 0x1000 | x | 00 | NT | T | 10 |
| b_1 | 6 | 0 | 0x1000 | 0 | 00 | NT | x | 00 |
| b_3 | 6 | 0 | 0x1004 | 0 | 00 | NT | T | 10 |
| b_0 | | | | | | | | |
| b_1 | | | | | | | | |
| b_3 | | | | | | | | |
| b_0 | | | | | | | | |
| b_1 | | | | | | | | |
| b_3 | | | | | | | | |
| b_0 | | | | | | | | |
| b_1 | | | | | | | | |
| b_3 | | | | | | | | |
| b_0 | | | | | | | | |
| b_1 | | | | | | | | |
| b_3 | | | | | | | | |
| b_0 | | | | | | | | |
| b_1 | | | | | | | | |
| b_3 | | | | | | | | |
| b_0 | | | | | | | | |
| b_1 | | | | | | | | |
| b_3 | | | | | | | | |

3. Parallel programming [10 points] **(Please write your answers directly on the exam paper)**

The following code is designed to implement a producer-consumer relationship between two independent threads executing on different processors. These threads share memory, and the memory system is sequentially consistent.

Assume the variables **`ring_buffer`**, **`head_index`**, **`tail_index`**, and **`number_in_buffer`** are shared between the threads. The statements in the provided code execute in order, but any given statement may not be atomic.

The **`SendData`** function puts one value into the communication buffer, and the **`ReceiveData`** function removes one value from the buffer. The first thread only calls **`SendData`**, and the second thread only calls **`ReceiveData`**.

Assume that there is a library which implements a mutual exclusion lock (mutex) which have two functions **`AcquireLock(int * lock)`** and **`ReleaseLock(int * lock)`**. Both **`AcquireLock`** and **`ReleaseLock`** take a parameter which is the address of a lock variable. Also, the lock library has a function **`InitializeLock(int * lock)`** which initializes a lock in the unlocked state when given a parameter which is the address of a lock.

For proper operation, does the following code need locking? If so, add the minimal **`AcquireLock, ReleaseLock, InitializeLock`**, and lock variable instantiations as needed. Ensure that the lock is held only for the minimal necessary time. Do not reorder the code below. Feel free to edit in-place, but please make your changes clear.

```
#define BUFFER_SIZE    4
int ring_buffer[BUFFER_SIZE];
int head_index = 0;
int tail_index = 0;
int number_in_buffer = 0;
void SendData(int data)
{
   while (number_in_buffer < BUFFER_SIZE) {
   }
   number_in_buffer = number_in_buffer + 1;
   ring_buffer[tail_index] = data;
   tail_index = (tail_index + 1) % BUFFER_SIZE;
}
int ReceiveData()
{
   int the_data;
   while (number_in_buffer <= 0) {
   }
   number_in_buffer = number_in_buffer - 1;
   the_data = ring_buffer[head_index];
   head_index = (head_index + 1) % BUFFER_SIZE;
   return the_data;
}
int main(void)
{
   LaunchThreads();
   return 0;
}
```

4. Out-of-Order execution.[10 points] **(Please write your answers directly on the exam paper)**

Draw the optimal pipeline diagram for the following code executing on the IO2I processor as shown below.

The IO2I processor fetches instructions in-order, issues instructions out-of-order, writes-back results out-of-order, and commits instructions in-order. Assume the processor can fetch one instruction per cycle, decode one instruction per cycle, issue one instruction per cycle, writeback one result per cycle, and commit one instruction per cycle.

Assume full bypassing of values from the respective instruction completion stage to the Issue stage. Assume that pipeline X can execute branches and ALU operations, pipeline L executes loads, pipeline S executes stores, and pipeline Y can execute multiply operations.
Loads have a latency of two cycles and ALU operations have a latency of one cycle. Branches are resolved in X0. Multiply instructions have a latency of four cycles.

Additional assumptions:

- The machine has no branch delay slots and always predicts the fall-through path.
- The register file has only one write port.
- Use a lower-case 'i' to denote if an instruction enters the issue queue but does not immediately issue.
- Use a lower-case 'r' to denote if an instruction enters the reorder buffer but does not immediately commit.
- The issue queue can hold 16 instructions and starts empty.

Using these details, draw the optimal pipeline diagram, labeling the pipeline stages as shown in the provided figure.

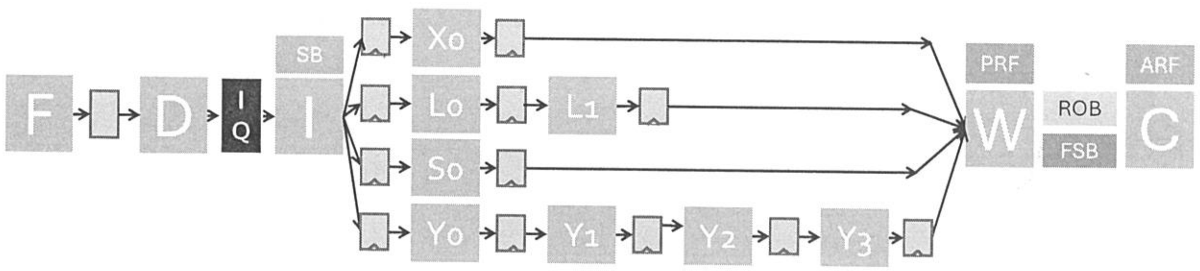


**Figure 2. IO2I Processor Pipeline**

**Code sequence**

```
0:  MUL  X6, X7, X8
1:  ADD  X9, X10, X11
2:  ADD  X11, X10, X6
3:  ADD  X13, X14, X15
4:  ADD  X19, X13, X11
5:  LW   X2, X3
6:  ADD  X12, X16, X19
7:  LW   X5, X2
8:  ADD  X15, X5, X12
```

| Instruction \ Timestamp | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | F | D | I | Y0 | Y1 | Y2 | Y3 | W | C | | | | |
| 1 | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | |

| Instruction \ Timestamp | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | |

5. Cache Coherence [10 points] **(Please write your answers directly on the exam paper)**

Consider a scenario with two processors using a snoopy MESI protocol. Each processor has a 2-line direct-mapped cache, with each line containing 16 bytes. Initially, all cache lines are marked as invalid.

Complete the following tables by indicating:

- Whether the processor experiences a cache hit or miss.
- Any bus transactions performed by the processor:
  - Bus Read Line (BRL)
  - Bus Write Line (BWL)
  - Bus Read-Invalidate Line (BRIL)
  - Bus Invalidate Line (BIL)
- For cache misses, specify the type of miss (compulsory, capacity, conflict, or coherence). A coherence miss is one where there would have been a hit, had some other processor not interfered.

Finally, provide the state of each processor's cache after all memory operations have been executed, in the order listed.

| Processor | Address | Read/Write | Cache Hit/Miss | Bus transaction(s) | "4C" miss type |
|---|---|---|---|---|---|
| 1 | 0x000 | Write | Miss | BRIL | Compulsory |
| 1 | 0x016 | Read | Miss | BRL | Compulsory |
| 2 | 0x010 | Read | | | |
| 1 | 0x006 | Write | | | |
| 2 | 0x018 | Write | | | |
| 1 | 0x150 | Write | | | |
| 1 | 0x230 | Write | | | |
| 1 | 0x010 | Read | | | |
| 2 | 0x000 | Read | | | |
| 1 | 0x150 | Read | | | |

Final state:

| | Proc 1 | |
|---|---|---|
| | Address | State |
| Set 0 | | |
| Set 1 | | |

| | Proc 2 | |
|---|---|---|
| | Address | State |
| Set 0 | | |
| Set 1 | | |

一一二學年度第二學期
博士班資格考

# 國立陽明交通大學 試 題 紙

科目：演算法 A

日期： 113 年 7 月 3 日 第 1 頁 共 1 頁

請 "✓" 明 ✓不可看書 可看書

* 請將答案依題號順序寫入答案卷

答題時字跡需工整，否則不予計分。Write your answers legibly; otherwise you will get zero score.

1. Greedy algorithm: Show that the activity-selection problem is solved by using the greedy approach. Definition of the activity-selection problem: We have a set of $S = \{a_1, a_2, \ldots a_n\}$ of $n$ proposed activities that wish to use a resource (can accommodate one activity only). Each activity $a_i$ has a start time $s_i$ and a finish time $t_i$. We wish to select a maximum-size subset of $S$. *15%*

2. (a) queue operations enQueue: insert an item into the tail of a queue and deQueue: remove an item from the head of a queue. Describe the algorithm to implement enQueue and deQueue using two stacks. *10%*

   (b) Show that enQueue and deQueue are done in constant amortized cost. *10%*

3. Given a set of real numbers $S = \{s_1, s_2, \ldots, s_n\}$, we wish to know is there $s_i$ and $s_j$ such that $|s_i - s_j| = d$. Design a divide and conquer algorithm to solve this problem. *15%*

# 國立陽明交通大學 試 題 紙

一一二學年度第二學期
博士班資格考

科目：演算法 B　　　　日期：113 年 7 月 3 日　第 1 頁 共 3 頁

請 "✓" 明　　✓不可看書　　可看書

* 請將答案依題號順序寫入答案卷

答題時字跡需工整，否則不予計分。Write your answers legibly; otherwise you will get zero score.

1. **[15%]** A[0 ~ N-1] and B[0 ~ N-1] are two strictly increasing arrays of N integers each; that is, A[i] < A[j] if i < j and B[i] < B[j] if i < j, where $0 \le i, j \le N-1$.
Your task is to find the N-th smallest element within these two arrays of N integers each (thus a total of 2N integers).

**Assume that `P` (initialized to 0) is the cumulative points you will earn on this problem.**

Note that, in the following, when you are asked to implement a **Θ**(N log N) algorithm, it should be just "N log N", not better and not worst than that. In your pseudo-code, you can use ***qsort***, ***quicksort***, ***mergesort***, ***insertion-sort*** or ***selection-sort*** instead of handcrafting them from scratch.

**(a)** Are you able to implement a **Θ**(N log N) algorithm to solve this problem?
If your answer is NO, then `P=P+2` and you are done (go to **(g)**, the end).
Else (if your answer is YES), then `P=P+2` and go to **(b)**.

**(b)** Please provide the pseudo-code of your **Θ**(N log N) algorithm.
If your code is considered nonsense, then `P=P-2`.
Else (if your code is not considered nonsense), then `P=P+3` or maybe `P=P+1` (partial credit).
Go to **(c)**.

**(c)** Are you able to implement a **Θ**(N) algorithm to solve this problem?
If your answer is NO, then `P=P+2` and you are done (go to **(g)**, the end).
Else (if your answer is YES), then `P=P+2` and go to **(d)**.

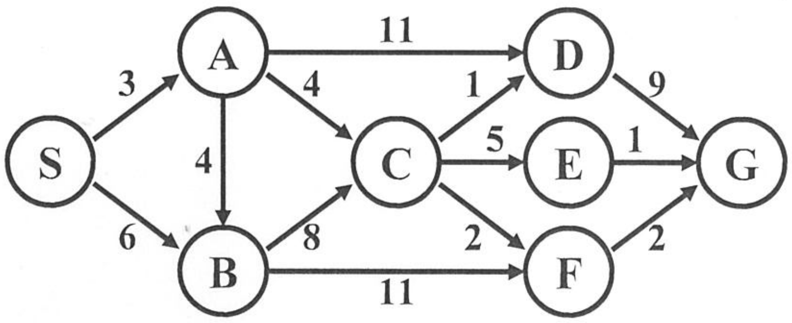
**(d)** Please provide the pseudo-code of your **Θ**(N) algorithm.
If your code is considered nonsense, then `P=P-2`.
Else (if your code is not considered nonsense, then `P=P+3` or maybe `P=P+1` (partial credit).
Go to **(e)**.

**(e)** Are you able to implement a **Θ**(log N) algorithm to solve this problem?
If your answer is NO, then `P=P+2` and you are done (go to **(g)**, the end).
Else (if your answer is YES), then `P=P+2` and go to **(f)**.

**(f)** Please provide the pseudo-code of your **Θ**(log N) algorithm.
If your code is considered nonsense, then `P=P-2`.
Else (if your code is not considered nonsense, then `P=P+3` or maybe `P=P+1` (partial credit).
Go to **(g)**.

**(g)** End – **you can earn up to 15 points!**

2. Given the following graph where vertex **S** is the source and the weight of each edge is specified alongside the edge, please apply the ***Bellman-Ford*** algorithm to find shortest paths from **S** to each of other vertices. During each pass/iteration of edge-by-edge relaxation, the order of edge to be processed is: **AB, AC, AD, BC, BF, CD, CE, CF, DG, EG, FG, SA, SB** (i.e., lexicographic order, assuming that an edge from vertex **X** to vertex **Y** is named **XY**).



There are 8 vertices in the graph; thus, in theory, the Bellman-Ford algorithm requires up to 7 passes of edge-by-edge relaxation to reach the steady state where correct shortest paths can be found. However, if you are lucky, it sometimes requires much fewer passes.

***** DO THIS *****

**(a)** **[7%]** Based on the aforementioned order of edge to be processed, please duplicate the graph for each pass of edge-by-edge relaxation and for each vertex **X** in the graph, derive ***d*[X]** where ***d*[X]** denotes the shortest-path weight from **S** to **X**. Duplicate as many as you want; you can stop when you reach the steady state (and the final correct value of ***d*[X]** for any vertex **X** can be found).

**(b)** **[3%]** In this particular graph and based on whatever bad/poor/stupid order of edge to be processed, is it possible that the Bellman-Ford algorithm requires the worst-case 7 passes to reach the steady state? Why or why not?

**(c)** **[3%]** In this particular graph, is it possible to have a "special" order of edge to be processed such that only 1 pass is required? If your answer is YES, what is the special order? If your answer is NO, why not?

一一二學年度第二學期
博士班資格考

# 國立陽明交通大學 試 題 紙

科目：演算法 B

日期： 113 年 7 月 3 日

3. Consider the same graph in the previous problem; but this time you are asked to find <u>longest</u> paths.

   *** **DO THIS** ***

   (a) **[5%]** Please briefly describe how you can modify the Bellman-Ford algorithm to solve the problem of finding longest paths.
   **HINT: You may consider modifying the weights of the edges in the graph.**

   (b) **[7%]** Please duplicate the graph <u>for each pass</u> of edge-by-edge relaxation and <u>for each vertex</u> **X** in the graph, derive ***d*[X]** where ***d*[X]** denotes the <u>longest</u>-path weight from **S** to **X**. Duplicate as many as you want; you can stop when you reach the steady state (and the final correct value of ***d*[X]** for any vertex **X** can be found).
   Order of edge to be processed: **AB, AC, AD, BC, BF, CD, CE, CF, DG, EG, FG, SA, SB** (same as specified in the previous problem)
   **NOTE: If you want, you can modify the weights of the edges as long as you clearly specify.**

4. Maximum (circular) subarray problem
   Input: A[0 ~ N-1] is an array of N integers
   Output: X and Y ($0 \le X \le Y \le N-1$) such that the sum of elements in subarray A[X ~ Y] is maximized

   *** **DO THIS** ***

   (a) **[5%]** Please provide the pseudo-code of an optimal algorithm with time complexity of **O**(N). Please use the following format:

```
=====
int MSP(A, N) {
  // Find X and Y!
  ...
  ...
  ...
  sum ← add up the elements in A[X ~ Y];
  return sum; // You only need to return the sum!
}
=====
```

   (b) **[5%]** Do the same as in **(a)**, i.e., pseudo-code, assuming that A[0 ~ N-1] is a circular array; that is, you may have $0 \le \mathbf{Y} \le \mathbf{X} \le$ N-1 and the maximum subarray is A[X ~ N-1] circularly followed by A[0 ~ Y]. Note that $0 \le \mathbf{X} \le \mathbf{Y} \le$ N-1 and a maximum subarray of A[X ~ Y] (non-circularly) are still possible.
   **HINT: Transform the original problem (by modifying A[0 ~ N-1]) and then reuse MSP(•, •) to solve the new problem. Go ahead if you want to call MSP(•, •) more than once!**