FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110328 DATA STRUCTURE AND ALGORITHM
Year I, First Semester, Final Examination, Dec 1, 2023 11:00-16:00

ชื่อ-นามสกุล.....เลขประจำตัว.....ตอนเรียนที่.....เลขที่ใน CR58.....

<u>หมายเหตุ</u>

1. ข้อสอบมีทั้งหมด 5 ข้อ ในกระดาษคำถามคำตอบ 9 หน้า
2. **ไม่อนุญาตให้นำตำราและเอกสารใดๆ เข้าในห้องสอบ**
3. **ไม่อนุญาตให้ใช้เครื่องคำนวณใดๆ**
4. ห้ามการหยิบยืมสิ่งใดๆ ทั้งสิ้น จากผู้สอบอื่นๆ เว้นแต่เจ้าหน้าที่ควบคุมการสอบจะหยิบยืมให้
5. ห้ามนำส่วนใดส่วนหนึ่งของข้อสอบและสมุดคำตอบออกจากห้องสอบ
6. ผู้เข้าสอบสามารถออกจากห้องสอบได้ หลังจากผ่านการสอบไปแล้ว 45 นาที
7. เมื่อหมดเวลาสอบ ผู้เข้าสอบต้องหยุดการเขียนใดๆ ทั้งสิ้น
8. **นิสิตกระทำผิดเกี่ยวกับการสอบ ตามข้อบังคับจุฬาลงกรณ์มหาวิทยาลัย มีโทษ คือ พ้นสภาพการเป็นนิสิต หรือ ได้รับสัญลักษณ์ F ในรายวิชาที่กระทำผิด และอาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้**

ห้ามนิสิตพกโทรศัพท์และอุปกรณ์สื่อสารไว้กับตัวระหว่างสอบ หากตรวจพบจะถือว่านิสิตกระทำผิดเกี่ยวกับการสอบ อาจต้องพ้นสภาพการเป็นนิสิต หรือ ให้ได้รับ F และอาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้

*** ร่วมรณรงค์การไม่กระทำผิดและไม่ทุจริตการสอบ ***

ข้าพเจ้ายอมรับในข้อกำหนดที่กล่าวมานี้ ข้าพเจ้าเป็นผู้ทำข้อสอบนี้ด้วยตนเองโดยมิได้รับการช่วยเหลือ หรือให้ความช่วยเหลือ ในการทำข้อสอบนี้

ลงชื่อนิสิต.....
วันที่.....

- ใช้ดินสอเขียนคำตอบได้
- ให้เขียนเลขที่ในใบเซ็นชื่อเข้าสอบทุกหน้า
- หากพื้นที่สำหรับเขียนคำตอบไม่เพียงพอ ให้เขียนไว้ด้านหลังของหน้านั้น ห้ามเขียนข้ามไปหน้าอื่น และให้ระบุไว้ในพื้นที่สำหรับเขียนคำตอบว่า “มีต่อด้านหลัง”

Part I ให้เติมคำลงในช่องว่างที่กำหนดไว้

1. (5 คะแนน) จงวิเคราะห์เวลาการทำงานของฟังก์ชันข้างล่างนี้ (ในรูปของ Θ ของฟังก์ชัน *n*)

`int method1(int n) {`
`  s = 0;`
`  for (int i=1; i<=n; i++) s++;`
`  for (int i=n; i>=1; i--) s++;`
`  return s;`
`}`
	method1 ใช้เวลา เป็น Θ(?)

`int method2(vector<int> &a){`
`  int s =0;`
`  for (int i = 0; i < a.length ; i++)`
`  for (int j = i; j >= 0 ; j--)`
`  s = a[i] – a[j];`
`  return s`
`}`
	method2 ใช้เวลา เป็น Θ(?)
	(ให้ n คือขนาดของ a

`int method3(int n) {`
`  if (n>0) {`
`  method3(n-3);`
`  method3(n-3);`
`  }`
`}`
	method3 ใช้เวลา เป็น Θ(?)

`method4(int n) {`
`  s = 0;`
`  for (int i=1; i<=n; i++)`
`  for (int j=1; j<=n; j++)`
`  if (i==j)`
`  for (int k = 0; k < n; k++)`
`  s++;`
`  return s;`
`}`
	method4 ใช้เวลา เป็น Θ(?)

`int method5(int n, int x) { //  0 <= x < n`
`  int sum = 0;`
`  for (int i = 0; i < n; i++) {`
`  sum = sum + i;`
`  int f = 0;`
`  while(f < n) { sum += f++; }`
`  if (i == x)  return sum;`
`  }`
`}`
	method5 ใช้เวลา เป็น Θ(?)

2. (4 คะแนน) สมมติให้ AVL Tree มีข้อมูลอยู่ภายในจำนวน 4 ตัว คือ 1, 2, 3, 4

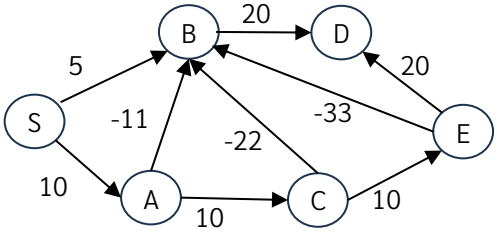
2.1. มี AVL Tree กี่ต้นที่แตกต่างกันที่มีข้อมูลดังกล่าว _____

2.2. จงวาด AVL Tree ทุกรูปแบบที่เป็นไปได้ (ให้ระบุค่าในปมด้วย)

3. (4 คะแนน) จากส่วนของโปรแกรมต่อไปนี้ จงวาด Recursion Tree เมื่อมีการเรียกฟังก์ชัน t(10)

```
void test(int n) {
  if (n > 0) {
    if (n % 2 == 0) {
      test(n/2);
      test(n/2);
    } else {
      test(n-1)
    }
  }
}
```

4. (5 คะแนน) จากกราฟด้านขวานี้ เราต้องการหา shortest path จากปม s โดยใช้ algorithm ของ Bellman-Ford ให้ D(a,b) คือระยะทางสั้นสุดจากปม s ไปยังปม b ที่ใช้เส้นเชื่อมไม่เกิน a จงเติมค่าของ D(a,b) ลงในตารางข้างล่างนี้



	S	A	B	C	D	E
0	0	∞	∞	∞	∞	∞
1						
2						
3						
4						
5						

Part II คำถามแบบตัวเลือก (เขียนคำตอบลงในกระดาษคำตอบเท่านั้น)

ในข้อย่อยต่อไปนี้ให้ตอบโดยเลือกคำตอบที่ถูกต้องที่สุด แต่ละข้อย่อยมีคะแนน 1 คะแนน หากไม่ตอบในข้อใด จะได้คะแนน 0 แต่ถ้าหากตอบผิดในข้อใด จะได้คะแนน -0.5 ต่อข้อย่อย อย่างไรก็ตาม ถึงแม้จะตอบผิดจนได้คะแนนรวมติดลบ จะถือว่าข้อนี้ได้คะแนนเป็น 0 ****ให้เขียนคำตอบลงในกระดาษคำตอบเท่านั้น****

ส่วนวิชา Data Structure

5. จงตอบคำถามต่อไปนี้

5.1. ความซับซ้อนของเวลาในการเข้าถึงข้อมูลใน C++ std::vector โดยใช้ operator[] คืออะไร?
- ก. O(1)
- ข. O(log n)
- ค. O(n)
- ง. O(n log n)

5.2. ความซับซ้อนของเวลาในการเข้าถึงข้อมูลใน C++ std::map โดยใช้ operator[] คืออะไร?
- ก. O(1)
- ข. O(log n)
- ค. O(n)
- ง. O(n log n)

5.3. ให้ s เป็น std::set<int> และให้ it เป็น iterator ที่ชี้ไปยังข้อมูลบางตัวของ s คำสั่งใดต่อไปนี้ทำให้ it ถูกทำให้ไม่ valid อีกต่อไป?
- ก. s.begin()
- ข. s.insert(30)
- ค. s.size()
- ง. ไม่มีข้อถูก

5.4. ข้อใดที่เป็นการเริ่มต้นใช้งาน vector ของจำนวนเต็ม 10 ตัว โดยแต่ละตัวถูกกำหนดค่าเป็น 5 อย่างถูกต้อง?
- ก. std::vector<int> v(10, 5);
- ข. std::vector<int> v = {10, 5};
- ค. std::vector<int> v(5, 10);
- ง. std::vector<int> v{10, 5};

5.5. ข้อใดเป็นความจริงเกี่ยวกับ iterators ของ vector หลังจากมีการจัดสรรพื้นที่ใหม่?
- ก. Iterator ทั้งหมดยังคงใช้งานได้
- ข. เฉพาะ iterator end() ที่ยังใช้งานได้
- ค. Iterator ทั้งหมดไม่สามารถใช้งานได้
- ง. เฉพาะ iterator begin() ที่ยังใช้งานได้

5.6. ในกรณีใดที่ iterator ของ vector ใน C++ จะถูกทำให้ไม่ valid อีกต่อไป?
- ก. การใช้ push_back() เมื่อ vector มีความจุเพียงพอ
- ข. การใช้ resize() เพื่อลดขนาดของ vector
- ค. การใช้ reserve() เพื่อเพิ่มความจุของ vector
- ง. การใช้ clear() เพื่อลบข้อมูลทั้งหมดออกจาก vector

5.7. ปัญหาที่อาจเกิดขึ้นจากการเพิ่มขนาดของ vector ใน C++ ด้วยการเพิ่มเล็กน้อยบ่อยครั้งโดยใช้ push_back() คืออะไร?
- ก. อาจนำไปสู่การรั่วของหน่วยความจำ
- ข. อาจทำให้เกิดการจัดสรรหน่วยความจำใหม่บ่อยครั้ง, นำไปสู่ปัญหาประสิทธิภาพ
- ค. อาจทำให้ข้อมูลใน vector ไม่สามารถถูกลบได้
- ง. เพิ่มความซับซ้อนของเวลาในการเข้าถึงข้อมูลเป็น O(n)

5.8. การดำเนินการใดที่ไม่ได้รับการสนับสนุนโดยตรงจาก std::stack ใน C++?
- ก. push()
- ข. pop()
- ค. top()
- ง. at()

5.9. คุณตรวจสอบจำนวนข้อมูลใน stack ของ C++ ได้อย่างไร?
- ก. stack.size()
- ข. stack.length()
- ค. stack.count()
- ง. stack.capacity()

5.10. เมธอดใดที่ใช้เพื่อลบข้อมูลบนสุดออกจาก stack ใน C++?
- ก. erase()
- ข. delete()
- ค. pop()
- ง. remove()

5.11. ปัญหาที่อาจเกิดขึ้นเมื่อคัดลอก std::stack ที่มีจำนวนข้อมูลจำนวนมากคืออะไร?
- ก. การทำสำเนา stack ไม่ได้รับอนุญาตใน C++
- ข. อาจนำไปสู่ปัญหาเรื่องประสิทธิภาพเนื่องจากการมีข้อมูลจำนวนมาก
- ค. ข้อมูลใน stack ใหม่จะอยู่ในลำดับที่กลับด้าน
- ง. การคัดลอก stack อาจทำให้ข้อมูลใน stack เดิมเสียหาย

5.12. ในการตรวจสอบเครื่องหมายวงเล็บของข้อมูล ([ ] { [ ] ( ) { { ( ) } โดยใช้ stack นั้น เมื่อเราตรวจพบว่ามีข้อผิดพลาดขึ้น ใน stack ยังมีข้อมูลอยู่กี่ตัว
- ก. 2
- ข. 3
- ค. 4
- ง. 5

5.13.การดำเนินการใดที่สามารถทำได้กับ queue ใน C++?
- ก. เข้าถึงข้อมูลตรงกลาง
- ข. เข้าถึงข้อมูลแบบสุ่ม
- ค. เพิ่มข้อมูลที่ด้านหลัง
- ง. การเรียงลำดับข้อมูล

5.14.ข้อใดเป็นความจริงเกี่ยวกับขนาดของ queue ใน C++ หลังจากเรียก queue::pop()?
- ก. ขนาดเป็นศูนย์เสมอ
- ข. ขนาดยังคงเหมือนเดิมหาก queue เป็นว่าง
- ค. ขนาดลดลงหนึ่ง ยกเว้นถ้า queue เป็นว่าง
- ง. ขนาดเพิ่มขึ้นหนึ่ง

5.15.ในการทำงานเกี่ยวกับการเปลี่ยนขนาดของคิววงกลมแบบไดนามิกที่ใช้ array, ปัจจัยสำคัญที่ควรพิจารณาในการดำเนินการ resize คืออะไร?
- ก. หัวของคิวต้องถูกจัดให้อยู่ตรงกับจุดเริ่มต้นของ array เสมอ
- ข. ข้อมูลต้องถูกจัดเรียงใหม่เพื่อรักษาลำดับแบบวงกลมใน array ใหม่
- ค. ต้องรีเซ็ตดัชนีท้ายหลังจากการเปลี่ยนขนาด
- ง. ขนาดสูงสุดของคิวควรเพิ่มเป็นสองเท่าในทุกๆ การเปลี่ยนขนาด

5.16.ในการวิเคราะห์ความซับซ้อน, สัญลักษณ์ Omega (Ω) หมายถึงอะไร?
- ก. ขอบเขตล่างของเวลาทำงานของอัลกอริทึม
- ข. ขอบเขตบนของเวลาทำงานของอัลกอริทึม
- ค. เวลาทำงานที่แน่นอนของอัลกอริทึม
- ง. เวลาทำงานเฉลี่ยของอัลกอริทึม

5.17.ความซับซ้อนของเวลาในการแทรกข้อมูลลงในอาร์เรย์ที่เรียงลำดับแล้วของ n ข้อมูลคืออะไร?
- ก. O(1)
- ข. O(log n)
- ค. O(n)
- ง. O(n log n)

5.18.ความซับซ้อนของเวลาสำหรับการเดินทางผ่านต้นไม้ทวิภาคแบบสมดุล (in-order traversal) ที่มีโหนด n โหนดคืออะไร?
- ก. O(n)
- ข. O(log n)
- ค. O(n log n)
- ง. O(n^2)

5.19.ความซับซ้อนของเวลาในการเพิ่มโหนดใหม่ที่ต้นของ linked list คืออะไร?
- ก. O(1)
- ข. O(log n)
- ค. O(n)
- ง. O(n^2)

5.20.ความซับซ้อนของเวลาในการเข้าถึงโหนดที่ n ใน singly linked list คืออะไร?
- ก. O(1)
- ข. O(log n)
- ค. O(n)
- ง. O(n log n)

5.21.ใน doubly linked list, ถ้าคุณมี pointer ไปยัง node หนึ่ง (ไม่ใช่หัวหรือท้ายของ list), จำนวนการอัพเดต pointer ขั้นต่ำที่จำเป็นในการลบ node นี้คือเท่าไร?
- ก. 1
- ข. 2
- ค. 4
- ง. 6

5.22.ใน circular doubly linked list, ปมหัว และ ปมท้าย ถูกเชื่อมโยงกันอย่างไร?
- ก. pointer ก่อนหน้าของหัวและ pointer ถัดไปของท้ายเป็น NULL.
- ข. pointer ก่อนหน้าของหัวชี้ไปที่ท้าย, และ pointer ถัดไปของท้ายชี้ไปที่หัว.
- ค. หัวและท้ายไม่ได้ถูกเชื่อมโยงโดยตรง แต่ถูกระบุด้วย pointer แยกต่างหาก.
- ง. list ไม่มีหัวและท้ายในแบบที่เข้าใจโดยทั่วไป.

5.23.ข้อใดอธิบาย max-heap ได้ดีที่สุด?
- ก. Binary heap ที่มีค่าของแต่ละโหนดมากกว่าหรือเท่ากับค่าของลูกๆ ของมัน
- ข. Binary heap ที่มีค่าของแต่ละโหนดน้อยกว่าหรือเท่ากับค่าของลูกๆ ของมัน
- ค. Binary heap ที่อนุญาตให้มีค่าซ้ำ
- ง. Binary heap ที่ไม่อนุญาตให้มีค่าซ้ำ

5.24.ความซับซ้อนของเวลาในการแทรกข้อมูลใน binary heap คืออะไร?
- ก. O(1)
- ข. O(log n)
- ค. O(n)
- ง. O(n log n)

5.25.ใน min-heap, ข้อมูลที่มีขนาดเล็กที่สุดจะพบได้ที่ไหน?
- ก. ที่โหนดใบซ้ายสุด
- ข. ที่โหนดใบขวาสุด
- ค. ที่โหนดราก
- ง. ที่โหนดใบใดๆ

5.26.ข้อใดเป็นกรณีการใช้งานทั่วไปสำหรับ binary heap?
- ก. เรียงลำดับข้อมูลตามลำดับเพิ่มขึ้น
- ข. การใช้งานเป็น priority queues
- ค. การสมดุลย์ binary search trees
- ง. การเก็บข้อมูลในลักษณะที่ไม่เป็นเส้นตรง

5.27.สำหรับ binary heap ที่ถูกแทนที่ด้วย array, ความซับซ้อนของเวลาสำหรับการดำเนินการ fixdown ที่โหนดที่มีความสูง h คืออะไร?
- ก. O(log n)
- ข. O(n)
- ค. O(h)
- ง. O(1)

5.28.คุณสมบัติพื้นฐานของ Binary Search Tree (BST) คืออะไร?
- ก. แต่ละโหนดมีลูกได้มากที่สุดสองตัว
- ข. ลูกทางซ้ายของโหนดมีค่ามากกว่าโหนดเสมอ
- ค. ลูกทางขวาของโหนดมีค่าน้อยกว่าโหนดเสมอ
- ง. ลูกทางซ้ายของโหนดมีค่าน้อยกว่าหรือเท่ากับโหนด และลูกทางขวามีค่ามากกว่าหรือเท่ากับโหนด

5.29.ข้อใดเป็นความจริงเกี่ยวกับการเดินทางแบบ inorder ของ BST?
- ก. มันสร้างรายการที่เรียงลำดับค่าจากมากไปน้อยเสมอ
- ข. มันสร้างรายการที่เรียงลำดับค่าจากน้อยไปมากเสมอ
- ค. มันได้ผลลัพธ์เป็นรายการที่แต่ละข้อมูลเป็นผลรวมของลูกๆ
- ง. มันให้ค่าในลำดับที่ถูกใส่เข้าไปในต้นไม้

5.30.ความซับซ้อนของเวลาในการค้นหาค่าใน BST ที่สมดุลคืออะไร?
- ก. O(1)
- ข. O(log n)
- ค. O(n)
- ง. O(n log n)

5.31.AVL tree คืออะไร?
- ก. Binary search tree ที่ความต่างของความสูงระหว่าง subtree ซ้ายและขวาของโหนดใดๆ มีไม่เกินหนึ่ง
- ข. Binary tree ที่แต่ละโหนดมีลูกได้ไม่เกินสาม
- ค. Binary search tree ที่แต่ละโหนดมีจำนวนโหนดเท่ากันใน subtree ซ้ายและขวา
- ง. Complete binary tree ที่เป็น binary search tree ด้วย

5.32.Balance Value ของโหนดใน AVL tree แสดงถึงอะไร?
- ก. ความต่างของจำนวนโหนดใน subtree ซ้ายและขวา
- ข. ความต่างของความสูงระหว่าง subtree ซ้ายและขวา
- ค. จำนวนโหนดทั้งหมดใน subtree ที่มีรากที่โหนดนั้น
- ง. ความลึกของโหนดในต้นไม้

5.33.วัตถุประสงค์ของการหมุนใน AVL tree คืออะไร?
- ก. เพื่อทำให้ต้นไม้เป็น complete binary tree
- ข. เพื่อให้แน่ใจว่าคุณสมบัติของ binary search tree ยังคงอยู่
- ค. เพื่อปรับสมดุลต้นไม้เมื่อเงื่อนไขความต่างของความสูงถูกละเมิด
- ง. เพื่อเพิ่มความสูงของต้นไม้เพื่อประสิทธิภาพในการค้นหาที่ดีขึ้น

5.34.ให้ a เป็น cp::map_avl และ b เป็น cp::map_bst โดยที่ทั้ง a และ b มีข้อมูลเหมือนกัน ข้อใดผิด
- ก. ต้นไม้ของ a มีความสูงน้อยกว่า b เสมอ
- ข. ทั้ง a และ b ใช้พื้นที่หน่วยความจำเท่ากันเสมอ
- ค. ปมรากของ a และ b เก็บข้อมูลที่มีค่าเท่ากันเสมอ
- ง. ผิดทุกข้อ

5.35.ความซับซ้อนของเวลาในการแทรกใน AVL tree คืออะไร?
- ก. O(1)
- ข. O(log n)
- ค. O(n)
- ง. O(n log n)

ส่วนวิชา Algorithm

5.36.อัลกอริทึมการเรียงลำดับในข้อใดที่ไม่เสถียร (รักษาลำดับก่อนหลังของข้อมูลที่คีย์เท่ากัน)?
- ก. Merge Sort
- ข. Bubble Sort
- ค. Quick Sort
- ง. Insertion Sort

5.37.ความซับซ้อนของเวลาในการเรียงลำดับด้วย Heap Sort ในกรณีที่ดีที่สุด, ปานกลาง, และแย่ที่สุดคืออะไร?
- ก. O(n log n) ในทุกกรณี
- ข. O(n) ในกรณีที่ดีที่สุด, O(n log n) ในกรณีปานกลางและแย่ที่สุด
- ค. O(n^2) ในทุกกรณี
- ง. O(n log n) ในกรณีที่ดีที่สุด, O(n^2) ในกรณีแย่ที่สุด

5.38.ส่วนใดในอัลกอริทึม Quick Sort ที่มีบทบาทสำคัญกับความซับซ้อนของเวลาที่สุด?
- ก. การเลือก pivot element
- ข. ความลึกของการเรียกฟังก์ชันแบบเรียกตัวเอง
- ค. วิธีการแบ่ง sub-arrays
- ง. การใช้ array รองสำหรับเก็บผลลัพธ์ชั่วคราว

5.39.อัลกอริทึมการเรียงลำดับใดที่โดยมีประสิทธิภาพสูงที่สุดสำหรับการเรียงลำดับข้อมูลที่เก็บด้วย linked lists? (และหากต้องมีการเก็บรายการข้อมูลชั่วคราว ก็ต้องใช้ linked list ในการเก็บเช่นกัน)
- ก. Quick Sort
- ข. Heap Sort
- ค. Merge Sort
- ง. Bubble Sort

5.40.ข้อใดอธิบายถึงอัลกอริทึมการค้นหาแบบ Brute Force ได้ดีที่สุด?
- ก. อัลกอริทึมที่ลองทุกโซลูชั่นเพื่อหาโซลูชั่นที่ถูกต้อง
- ข. อัลกอริทึมค้นหาที่มีประสิทธิภาพสูงซึ่งใช้ heuristic ขั้นสูง
- ค. อัลกอริทึมที่แบ่งปัญหาออกเป็นปัญหาย่อย
- ง. อัลกอริทึมค้นหาที่ใช้การสุ่มเพื่อหาโซลูชั่น

5.41.ข้อจำกัดหลักของอัลกอริทึมการค้นหาแบบบรูตฟอร์ซคืออะไร?
- ก. ไม่สามารถนำมาใช้กับปัญหาที่ซับซ้อนได้
- ข. มีประสิทธิภาพเกินไปและอาจมองข้ามโซลูชั่นที่ดีกว่า
- ค. ต้องการพื้นที่หน่วยความจำมาก
- ง. อาจช้าสำหรับพื้นที่ปัญหาขนาดใหญ่

5.42.ลักษณะสำคัญของอัลกอริทึม divide and conquer คืออะไร?
- ก. การแบ่งปัญหาออกเป็นปัญหาย่อยขนาดเล็ก, แก้ไขแต่ละปัญหาอย่างอิสระ, และรวมผลลัพธ์เข้าด้วยกัน
- ข. การแก้ปัญหาโดยวิธีการวนซ้ำโดยใช้ลูปเดียว
- ค. การใช้ thread หลายตัวเพื่อแก้ไขส่วนต่างๆ ของปัญหาพร้อมกัน
- ง. การแบ่งข้อมูลเป็นสองส่วนเท่าๆ กันและจากนั้นจึงเรียงลำดับแต่ละส่วน

5.43.อัลกอริทึมแบบ divide and conquer ที่มีสมการการเกิดซ้ำ T(n) = 2T(n/2) + n^2 ความซับซ้อนของเวลาของอัลกอริทึมนี้ตาม Master Theorem คืออะไร?
- ก. O(n^2)
- ข. O(n^2 log n)
- ค. O(n log n)
- ง. O(n^3)

5.44.ในอัลกอริทึมแบบ divide and conquer สำหรับปัญหาคู่จุดที่ใกล้ที่สุด, ขั้นตอนสำคัญที่ดำเนินการหลังจากแบ่งจุดคืออะไร?
- ก. เรียงจุดตามพิกัด x
- ข. การรวมครึ่งกลับมาในเวลาเชิงเส้น
- ค. ตรวจสอบคู่จุดที่ใกล้กันข้ามเส้นแบ่ง
- ง. คำนวณระยะทางใหม่สำหรับทุกคู่จุด

5.45.ความแตกต่างหลักระหว่าง dynamic programming กับวิธีการ divide and conquer คืออะไร?
- ก. dynamic programming แก้ปัญหาโดยวิธีการเรียกซ้ำเท่านั้น, ในขณะที่ divide and conquer ไม่ทำเช่นนั้น
- ข. divide and conquer แบ่งปัญหาออกเป็นปัญหาย่อยที่ไม่ซ้อนทับกัน, ในขณะที่ dynamic programming ใช้เมื่อปัญหาย่อยมีการซ้อนทับ
- ค. dynamic programming ไม่สามารถใช้สำหรับปัญหาที่เกี่ยวกับการหาค่าที่ดีที่สุด
- ง. การแก้ปัญหาโดยใช้ divide and conquer มักจะเร็วกว่าการใช้ dynamic programming เสมอ

5.46.คุณสมบัติใดที่จำเป็นสำหรับปัญหาที่จะถูกแก้ไขโดยใช้ dynamic programming?
- ก. Greedy Property
- ข. Optimal Substructure
- ค. Recursive Structure
- ง. Divisibility

5.47.เราจะสามาลดหน่วยความจำที่ต้องใช้ในการ ปัญหา 0/1 knapsack ด้วย dynamic programming ได้อย่างไร?
- ก. โดยใช้เพียง array เดียวในการเก็บสถานะปัจจุบันและสถานะก่อนหน้า
- ข. โดยการบีบอัดพื้นที่สถานะโดยใช้ฟังก์ชันแฮช
- ค. โดยการนำไปใช้แบบเรียกตัวเองโดยไม่มีตารางการจดจำ
- ง. โดยการแบ่งปัญหาออกเป็นปัญหาย่อยที่อิสระต่อกัน

5.48.ในปัญหา Matrix Chain Multiplication เป้าหมายหลักของวิธีการ dynamic programming คืออะไร?
- ก. การหาผลคูณของเมทริกซ์
- ข. การคำนวณผลคูณที่ใหญ่ที่สุดที่เป็นไปได้ของมิติเมทริกซ์
- ค. การกำหนดลำดับการคูณที่ลดจำนวนการคูณสเกล่าทั้งหมดให้น้อยที่สุด
- ง. การลดปัญหาลงเป็นการบวกเมทริกซ์แทนการคูณ

5.49.พิจารณาเกมที่ผู้เล่นสองคนผลัดเลือกตัวเลขจากปลายทั้งสองของอาร์เรย์ ความซับซ้อนของเวลาของโซลูชัน dynamic programming ในการหาผลรวมสูงสุดของตัวเลขที่เลือกที่เป็นไปได้สำหรับผู้เล่นคนแรกโดยสมมติว่าทั้งสองผู้เล่นเล่นอย่างดีที่สุดอะไร เมื่อตอนเริ่มมีอาร์เรย์มีตัวเลขอยู่ n ตัว?
- ก. O(2^n)
- ข. O(n^3)
- ค. O(n^2)
- ง. O(n)

5.50.ในปัญหาการเลือกกิจกรรม (activity selection problem), ทำไมอัลกอริทึมที่โลภของการเลือกกิจกรรมที่มีเวลาจบเร็วที่สุดจึงผลิตโซลูชันที่เหมาะสมที่สุดเสมอ?
- ก. เพราะมันทำให้เหลือพื้นที่มากที่สุดสำหรับกิจกรรมที่เหลือ
- ข. เพราะมันเลือกกิจกรรมที่มีระยะเวลาสั้นที่สุดเสมอ
- ค. เพราะมันทำให้จำนวนกิจกรรมที่ทำได้สูงสุดในเวลาที่น้อยที่สุด
- ง. เพราะกิจกรรมที่มีเวลาจบเร็วเสมอมีค่ามากกว่า

5.51.ปัญหาใดต่อไปนี้ไม่สามารถแก้ไขได้อย่างเหมาะสมโดยใช้อัลกอริทึมที่โลภ?

ก. การหา minimum spanning tree ของกราฟ
ข. การหาเส้นทางที่สั้นที่สุดในกราฟจากจุดเริ่มต้นไปยังทุกจุด
ค. ปัญหาการเปลี่ยนเหรียญสำหรับรายการเหรียญแบบใดก็ได้ที่กำหนด
ง. การจัดตารางงานบนเครื่องจักรเดี่ยวเพื่อลดความล่าช้าทั้งหมด

5.52.เหตุผลหลักที่ปัญหา knapsack แบบ fractional สามารถแก้ไขได้อย่างเหมาะสมโดยใช้อัลกอริทึมที่โลภแตกต่างจากปัญหา knapsack แบบ 0/1 อย่างไร?
ก. ของสามารถถูกแบ่งออกเป็นส่วนเล็กๆ ได้
ข. มีของจำนวนจำกัด
ค. อัตราส่วนกำไรต่อน้ำหนักของของแต่ละชิ้นเป็นค่าคงที่เสมอ
ง. ความจุของถุงแปรผันได้

5.53.คุณมีตาราง 2 มิติที่บางเซลล์มีสิ่งกีดขวาง คุณจะใช้อัลกอริทึมอะไรเพื่อหาจำนวนเส้นทางที่ไม่ซ้ำกันจากมุมบนซ้ายไปยังมุมล่างขวา โดยที่คุณสามารถเคลื่อนที่ได้เฉพาะลงหรือขวาเท่านั้น?
ก. ใช้ตาราง memoization เพื่อเก็บจำนวนวิธีในการเข้าถึงแต่ละเซลล์โดยพิจารณาอุปสรรค
ข. คำนวณจำนวนเส้นทางทั้งหมดโดยไม่มีอุปสรรค และหักล้างเส้นทางที่ผ่านอุปสรรค
ค. ใช้วิธีการเรียกซ้ำที่กลับมาทันทีเมื่อพบกับอุปสรรค
ง. ใช้อัลกอริทึมแบบละโมบเพื่อเลือกเส้นทางที่มีอุปสรรคน้อยที่สุดเสมอจากแต่ละช่องแล้วนำมาบวกกัน

5.54.ปัญหาการแบ่งส่วนถามว่าเซตที่กำหนดสามารถแบ่งออกเป็นสอง subset ได้หรือไม่ เพื่อให้ผลรวมขององค์ประกอบในทั้งสอง subset เท่ากัน วิธีการ dynamic programming ในการแก้ปัญหานี้คืออะไร?
ก. รวมทุกข้อมูลและตรวจสอบว่าผลรวมทั้งหมดเป็นเลขคี่หรือคู่
ข. แบ่งเซตเป็นสองส่วนแบบเรียกซ้ำและตรวจสอบว่าทั้งสองส่วนมีผลรวมเท่ากันหรือไม่
ค. เรียงลำดับเซตและใช้สองตัวชี้จากท้ายทั้งสองข้างเพื่อหาการแบ่ง
ง. ใช้ตาราง dynamic programming 2 มิติเพื่อติดตามผลรวมที่สามารถทำได้ด้วย subset ที่มีขนาดเพิ่มขึ้น

5.55.คุณสามารถตรวจจับ cycle ในกราฟระบุทิศทางได้อย่างไร?
ก. โดยการตรวจสอบว่ากราฟเชื่อมต่อกันหรือไม่
ข. โดยใช้การค้นหาแบบกว้างออกไป (BFS) และตรวจสอบ edge ทางข้าม
ค. โดยการตรวจสอบว่าทุกปมมีดีกรีเป็นคู่หรือไม่
ง. โดยใช้การค้นหาแบบลึกลงไป (DFS) และตรวจสอบ edge ย้อนกลับ

5.56.สถานการณ์ใดที่ topological ของกราฟระบุทิศทางเป็นประโยชน์?
ก. เมื่อหาเส้นทางที่สั้นที่สุดในกราฟ
ข. เมื่อค้นหา weakly connected component
ค. เมื่อพยายามตรวจจับ cycle ในกราฟที่ไม่มีทิศทาง
ง. เมื่อจัดตารางงานโดยที่งานเป็นปม และ ระบุว่างานใดต้องเสร็จก่อนงานใดด้วยเส้น

5.57.ในกราฟที่ไม่มีน้ำหนัก, BFS ช่วยหาเส้นทางที่สั้นที่สุดระหว่างสองปมได้อย่างไร?
ก. โดยการเลือกเส้นทางที่มีน้ำหนักน้อยที่สุดเสมอ
ข. โดยการสำรวจปมทั้งหมดในระดับปัจจุบันก่อนก่อนที่จะเคลื่อนไปยังระดับถัดไป
ค. โดยใช้คิวลำดับความสำคัญเพื่อเลือกปมถัดไปที่จะเยี่ยมชม
ง. โดยการติดตามย้อนกลับจากปมปลายทางไปยังปมต้นทาง

5.58.การใช้งานใดต่อไปนี้เป็นการใช้งานทั่วไปของ BFS ในทฤษฎีกราฟ?
ก. การหาส่วนประกอบที่เชื่อมต่อแบบแข็งแกร่ง
ข. การคำนวณ minimum spanning tree
ค. การหา component ที่เชื่อมต่อในกราฟที่ไม่มีทิศทาง
ง. การแก้ปัญหานักเดินทาง

5.59.คุณสมบัติสำคัญของเมทริกซ์ความเชื่อมโยง (Adjacency Matrix) ของกราฟที่ไม่มีทิศทางคืออะไร?
ก. เมทริกซ์เป็นเมทริกซ์ที่ผกผันได้เสมอ (Invertible)
ข. เมทริกซ์เป็นเมทริกซ์ที่สมมาตร (Symmetric)
ค. ทุกจุดในเมทริกซ์มีค่าเป็น 0 หรือ 1
ง. ผลรวมในแถวใดๆ มีค่าเท่ากับจำนวนปมในกราฟ

5.60.ในการแทนกราฟทิศทางด้วยเมทริกซ์ความเชื่อมโยง, ค่า 1 ที่ตำแหน่ง (i, j) บ่งบอกถึงอะไร?
ก. มีเส้นเชื่อมสองทางระหว่างปม i และ j
ข. มีเส้นเชื่อมจากปม j ไปยังปม i
ค. ปม i และ j ไม่เชื่อมต่อกัน
ง. มีเส้นเชื่อมจากปม i ไปยังปม j

5.61.ความซับซ้อนของพื้นที่ในการแทนกราฟที่มีปม n ปมด้วยเมทริกซ์ความเชื่อมโยงคืออะไร?
ก. O(n)
ข. O(n^2)
ค. O(log n)
ง. O(n log n)

5.62.ข้อความใดที่เปรียบเทียบอัลกอริทึม Kruskal และ Prim สำหรับการหา minimum spanning tree (MST) ได้อย่างถูกต้อง?
ก. อัลกอริทึม Kruskal มีประสิทธิภาพมากกว่าในกราฟหนาแน่น, ในขณะที่ Prim ดีกว่าสำหรับกราฟที่หายาก

ข. ทั้งสองอัลกอริทึมต้องการให้กราฟถูกเรียงลำดับตามน้ำหนักเส้นเชื่อมก่อนการทำงาน
ค. ทั้งสองอัลกอริทึมมีความซับซ้อนของเวลาเท่ากันในทุกกรณี
ง. อัลกอริทึม Kruskal สร้าง MST โดยการเพิ่มเส้นเชื่อม, ในขณะที่ Prim สร้าง MST โดยการเพิ่มปม

5.63.คุณสมบัติใดต่อไปนี้เป็นคุณสมบัติที่จริงของ minimum spanning tree ทุกต้นในกราฟ?
ก. MST ประกอบด้วยเส้นทางที่สั้นที่สุดระหว่างปมใดๆ สองจุดในกราฟ
ข. MST มีเส้นเชื่อมมากกว่าต้นไม้อื่นๆ ของกราฟ
ค. น้ำหนักรวมของ MST น้อยกว่าหรือเท่ากับน้ำหนักของต้นไม้อื่นๆ ในกราฟ
ง. MST สามารถพบได้เฉพาะในกราฟที่ไม่มีทิศทาง

5.64.อัลกอริทึม minimum spanning tree ใดที่ใช้โครงสร้างข้อมูล disjoint set เป็นหลัก?
ก. อัลกอริทึม Prim
ข. อัลกอริทึม Kruskal
ค. อัลกอริทึม Borůvka
ง. อัลกอริทึม Dijkstra

5.65.ข้อจำกัดใดต่อไปนี้เป็นข้อจำกัดของอัลกอริทึม Dijkstra สำหรับการหาเส้นทางที่สั้นที่สุด?
ก. ไม่สามารถใช้กับเส้นทางที่มีน้ำหนักเป็นลบ
ข. ไม่สามารถใช้กับกราฟที่ไม่มีทิศทาง
ค. ไม่สามารถหาเส้นทางที่สั้นที่สุดในกราฟที่มีวงจร
ง. สามารถคำนวณเส้นทางที่สั้นที่สุดจากจุดเริ่มต้นเดียวไปยังจุดหมายปลายทางเดียว

5.66.คุณลักษณะเฉพาะใดของอัลกอริทึม Bellman-Ford เมื่อเปรียบเทียบกับอัลกอริทึม Dijkstra ในการหาเส้นทางที่สั้นที่สุด?
ก. สามารถจัดการกับกราฟที่มีเส้นทางที่มีน้ำหนักเป็นลบ
ข. ทำงานเร็วกว่าอัลกอริทึม Dijkstra ในทุกกรณี
ค. ต้องการให้กราฟไม่มีวงจร
ง. ไม่จำเป็นต้องระบุจุดเริ่มต้น

5.67.ปัญหาใดที่อัลกอริทึม Floyd-Warshall แก้ไข?
ก. การหาเส้นทางที่สั้นที่สุดระหว่างทุกคู่ของปมในกราฟที่มีน้ำหนัก
ข. การกำหนด minimum spanning tree ของกราฟ
ค. การคำนวณเส้นทางที่สั้นที่สุดจากจุดเริ่มต้นเดียวไปยังปมอื่นๆ ทั้งหมด
ง. การหาเส้นทางที่ยาวที่สุดในกราฟทิศทางที่ไม่มีวงจร

5.68.ปัญหาเส้นทางที่สั้นที่สุดในกราฟทิศทางที่ไม่มีวงจร (DAG) มักจะถูกแก้ไขอย่างไร?
ก. โดยใช้เวอร์ชันที่ปรับปรุงของอัลกอริทึม Dijkstra ที่อนุญาตให้มีเส้นทางที่มีน้ำหนักเป็นลบ
ข. โดยการทำ topology sort ก่อน ตามด้วยการคำนวณระยะทางโดยใช้เวลา linear time
ค. โดยการประยุกต์ใช้อัลกอริทึม Bellman-Ford
ง. โดยการถือว่า DAG เป็นกราฟที่ไม่มีทิศทางและใช้อัลกอริทึม Dijkstra

5.69.ความซับซ้อนของเวลาของอัลกอริทึม Dijkstra เมื่อนำไปใช้งานกับ binary heap (priority queue) คืออะไร?
ก. O(V^2), โดยที่ V คือจำนวนของปม
ข. O((V + E) log V), โดยที่ V คือจำนวนปม และ E คือจำนวนเส้นเชื่อม
ค. O(V log V + E), โดยที่ V คือจำนวนปม และ E คือจำนวนเส้นเชื่อม
ง. O(E log V), โดยที่ E คือจำนวนเส้นเชื่อม และ V คือจำนวนปม

5.70.อัลกอริทึม Floyd-Warshall สามารถใช้ในการตรวจจับวงจรลบในกราฟได้หรือไม่?
ก. ได้, โดยการตรวจสอบค่าที่เป็นลบบนแนวทแยงของเมทริกซ์ระยะทางสั้นสุดระหว่างคู่ปม
ข. ไม่ได้, มันสามารถหาเฉพาะวงจรบวกเท่านั้น
ค. ได้, โดยการระบุเส้นทางที่มีน้ำหนักเป็นลบ
ง. ไม่ได้, มันไม่ได้ถูกออกแบบมาเพื่อตรวจจับวงจร