Kuis 1  
Struktur Data dan Algoritma  
Fakultas Ilmu Komputer Universitas Indonesia  
Durasi: 30 menit

**Petunjuk**: Tuliskan jawaban Anda sesingkat-singkatnya (**tanpa penjelasan apapun!**) di dalam kotak jawaban yang disediakan pada kolom paling kanan.

|  |  |  |
| --- | --- | --- |
| **No.** | **Soal** | **Jawaban** |
|  | Urutkan kompleksitas algoritma berikut ini, mulai dari yang paling rendah (paling kecil *growth rate*-nya) sampai yang paling tinggi (paling besar *growth rate*-nya).   * 1. O(N2 . log √N)   2. O(log(2N).log(10N))   3. O(log N3. log N2)   4. O(√N logN2)   5. O(N!)   6. O(2N (N/2)!) | **Tuliskan huruf-hurufnya saja:**  **Jawaban:  C, d, b, a, f, e** |
|  | Algoritma A dan B kompleksitasnya sama yaitu , tentukan apakah pernyataan berikut ini BENAR atau SALAH: | **BENAR/SALAH:** |
| * + - * 1. Untuk data yang kecil waktu eksekusi A pasti sama dengan B. | Salah |
| * + - * 1. Untuk data yang besar waktu eksekusi A pasti sama dengan B. | Salah |
| * + - * 1. Jika untuk data berukuran N yang besar waktu eksekusi A dua kali waktu eksekusi B, maka untuk ukuran data 2N, waktu eksekusi A akan sama dengan waktu eksekusi B. | salah |
| * + - * 1. Jika untuk data berukuran N yang besar waktu eksekusi A dua kali waktu eksekusi B, maka untuk ukuran data 2N, waktu eksekusi A akan sama dengan 4 kali waktu eksekusi B. | salah |
|  | Apakah kompleksitas waktu potongan kode di bawah ini dalam notasi Big-Oh?  for (int i = 1; i <= N; i \*= 3) {  for (int j = 1; j <= i\*N; j += i) {  System.out.println(i+” ”+j);  }  } | O(N log N) |
|  | Diberikan fungsi rekursif berikut:  int f(int n) {  if (n <= 1) return n;  else if (n <= 5) return f(n-2)+n;  else return f(n/2)+n;  }  Berapakah nilai f(25)? | 47 |
|  | Diberikan fungsi rekursif algoritma binary search sebagai berikut:  int binsearch(data[ ], x, low, high){  mid = (low+high) / 2;  if( data[mid] == x )  return mid;  else if ( x < data[mid] )  return binsearch(data[ ], x, low, mid );  else  return binsearch(data[ ], x, mid+1, high);  }  Jika a adalah sebuah *sorted array* berisi bilangan 1 sampai dengan 100 (berturut-turut), berapa kali *recursive call* yang terjadi pada pemanggilan binsearch(a, 80, 0, a.length-1)termasuk pemanggilan pertama ? | 7 kali |
|  | Apa output dari potongan program di bawah ini?  String[] values = {"Sumatera", "Kalimantan", "Jawa", "Sulawesi", "Bali", "Papua"};  Stack<String> stackSatu = new Stack<String> ();  Deque<String> queueDua = new ArrayDeque<String> ();    for(int i = 0; i < values.length; i++) {  if(i % 2 == 0) {  stackSatu.push(values[i]);  }  else {  queueDua.add(values[i]);  }  }    while(!stackSatu.empty()) {  String x = stackSatu.pop();  queueDua.add(x);  }    while(!queueDua.isEmpty()) {  System.out.println(queueDua.poll());  } | Kalimantan, Sulawesi, Papua, Bali, Jawa, Sumatera |
|  | Jika *swapping* adalah proses menukarkan isi dua elemen array, berapa kali dilakukan proses *swapping* oleh algoritma **Bubble-sort** jika dijalankan untuk mengurutkan (dari kecil ke besar) array berikut ini:  {10, 9, 8, 7, 6, 5, 4, 3, 2, 1} ? |  |

Kuis 1  
Struktur Data dan Algoritma  
Fakultas Ilmu Komputer Universitas Indonesia  
Durasi: 30 menit

**Petunjuk**: Tuliskan jawaban Anda sesingkat-singkatnya (**tanpa penjelasan apapun!**) di dalam kotak jawaban yang disediakan pada kolom paling kanan.

|  |  |  |
| --- | --- | --- |
| **No.** | **Soal** | **Jawaban** |
|  | Urutkan kompleksitas algoritma berikut ini, mulai dari yang paling rendah (paling kecil *growth rate*-nya) sampai yang paling tinggi (paling besar *growth rate*-nya).   * 1. O(√N logN2)   2. O(N!)   3. O(2N (N/2)!)   4. O(N2 . log √N)   5. O(log(2N).log(10N))   6. O(log N3. log N2) | **Tuliskan huruf-hurufnya saja:**  **f, a, e, d, c, b** |
|  | Algoritma A dan B kompleksitasnya sama yaitu , tentukan apakah pernyataan berikut ini BENAR atau SALAH: | **BENAR/SALAH:** |
| * + - * 1. Jika untuk data berukuran N yang besar waktu eksekusi A dua kali waktu eksekusi B, maka untuk ukuran data 2N, waktu eksekusi A akan sama dengan waktu eksekusi B. | Salah |
| * + - * 1. Jika untuk data berukuran N yang besar waktu eksekusi A dua kali waktu eksekusi B, maka untuk ukuran data 2N, waktu eksekusi A akan sama dengan 4 kali waktu eksekusi B. | Salah |
| * + - * 1. Untuk data yang kecil waktu eksekusi A pasti sama dengan B. | salah |
| * + - * 1. Untuk data yang besar waktu eksekusi A pasti sama dengan B. | salah |
|  | Apakah kompleksitas waktu potongan kode di bawah ini dalam notasi Big-Oh?  for (int i = 1; i <= N; i \*= 4) {  for (int j = 1; j <= i\*N; j += i) {  System.out.println(i+” ”+j);  }  } | O(N log N) |
|  | Diberikan fungsi rekursif berikut:  int f(int n) {  if (n <= 1) return n;  else if (n <= 5) return f(n-2)+n;  else return f(n/2)+n;  }  Berapakah nilai f(28)? | 53 |
|  | Diberikan fungsi rekursif algoritma binary search sebagai berikut:  int binsearch(data[ ], x, low, high){  mid = (low+high) / 2;  if( data[mid] == x )  return mid;  else if ( x < data[mid] )  return binsearch(data[ ], x, low, mid );  else  return binsearch(data[ ], x, mid+1, high);  }  Jika a adalah sebuah *sorted array* berisi bilangan 1 sampai dengan 100 (berturut-turut), berapa kali *recursive call* yang terjadi pada pemanggilan binsearch(a, 84, 0, a.length-1)termasuk pemanggilan pertama ? | 5 |
|  | Apa output dari potongan program di bawah ini?  String[] values = {"Sumatera", "Kalimantan", "Jawa", "Sulawesi", "Bali", "Papua"};  Stack<String> stackSatu = new Stack<String> ();  Deque<String> queueDua = new ArrayDeque<String> ();    for(int i = 0; i < values.length; i++) {  if(i % 2 == 0) {  stackSatu.push(values[i]);  }  else {  queueDua.add(values[i]);  }  }    while(!queueDua.isEmpty()) {  String x = queueDua.poll();  stackSatu.push(x);  }    while(!stackSatu.empty()) {  System.out.println(stackSatu.pop());  } | Papua, Sulawesi, Kalimantan, Bali, Jawa, SUmatera |
|  | Jika *swapping* adalah proses menukarkan isi dua elemen array, berapa kali dilakukan proses *swapping* oleh algoritma **Selection-sort** jika dijalankan untuk mengurutkan (dari kecil ke besar) array berikut ini:  {10, 9, 8, 7, 6, 5, 4, 3, 2, 1} ? | 5 atau 9 |

Kuis 1  
Struktur Data dan Algoritma  
Fakultas Ilmu Komputer Universitas Indonesia  
Durasi: 30 menit

**Petunjuk**: Tuliskan jawaban Anda sesingkat-singkatnya (**tanpa penjelasan apapun!**) di dalam kotak jawaban yang disediakan pada kolom paling kanan.

|  |  |  |
| --- | --- | --- |
| **No.** | **Soal** | **Jawaban** |
|  | Urutkan kompleksitas algoritma berikut ini, mulai dari yang paling rendah (paling kecil *growth rate*-nya) sampai yang paling tinggi (paling besar *growth rate*-nya).   * 1. O(log N3. log N2)   2. O(√N logN2)   3. O(N!)   4. O(2N (N/2)!)   5. O(N2 . log √N)   6. O(log(2N).log(10N)) | **Tuliskan huruf-hurufnya saja: a, b, f, e, d, c** |
|  | Algoritma A dan B kompleksitasnya sama yaitu , tentukan apakah pernyataan berikut ini BENAR atau SALAH: | **BENAR/SALAH:** |
| * + - * 1. Untuk data yang besar waktu eksekusi A pasti sama dengan B. | Salah |
| * + - * 1. Untuk data yang kecil waktu eksekusi A pasti sama dengan B. | Salah |
| * + - * 1. Jika untuk data berukuran N yang besar waktu eksekusi A dua kali waktu eksekusi B, maka untuk ukuran data 2N, waktu eksekusi A akan sama dengan 4 kali waktu eksekusi B. | Salah |
| * + - * 1. Jika untuk data berukuran N yang besar waktu eksekusi A dua kali waktu eksekusi B, maka untuk ukuran data 2N, waktu eksekusi A akan sama dengan waktu eksekusi B. | Salah |
|  | Apakah kompleksitas waktu potongan kode di bawah ini dalam notasi Big-Oh?  for (int i = 1; i <= N; i \*= 5) {  for (int j = 1; j <= i\*N; j += i) {  System.out.println(i+” ”+j);  }  } | O(N log N) |
|  | Diberikan fungsi rekursif berikut:  int f(int n) {  if (n <= 1) return n;  else if (n <= 5) return f(n-2)+n;  else return f(n/2)+n;  }  Berapakah nilai f(31)? | 57 |
|  | Diberikan fungsi rekursif algoritma binary search sebagai berikut:  int binsearch(data[ ], x, low, high){  mid = (low+high) / 2;  if( data[mid] == x )  return mid;  else if ( x < data[mid] )  return binsearch(data[ ], x, low, mid );  else  return binsearch(data[ ], x, mid+1, high);  }  Jika a adalah sebuah *sorted array* berisi bilangan 1 sampai dengan 100 (berturut-turut), berapa kali *recursive call* yang terjadi pada pemanggilan binsearch(a, 91, 0, a.length-1)termasuk pemanggilan pertama ? | 5 |
|  | Apa output dari potongan program di bawah ini?  String[] values = {"Sumatera", "Kalimantan", "Jawa", "Sulawesi", "Bali", "Papua"};  Deque<String> queueSatu = new ArrayDeque<String> ();  Stack<String> stackDua = new Stack<String> ();    for(int i = 0; i < values.length; i++) {  if(i % 2 == 0) {  queueSatu.add(values[i]);  }  else {  stackDua.push(values[i]);  }  }    while(!stackDua.empty()) {  String x = stackDua.pop();  queueSatu.add(x);  }    while(!queueSatu.isEmpty()) {  System.out.println(queueSatu.poll());  } | Sumatera, Jawa, Bali, Papua, Sulawesi, Kalimantan |
|  | Jika *swapping* adalah proses menukarkan isi dua elemen array, berapa kali dilakukan proses *swapping* oleh algoritma **Bubble-sort** jika dijalankan untuk mengurutkan (dari kecil ke besar) array berikut ini:  {10, 9, 8, 7, 6, 5, 4, 3, 2, 1} ? | 45 |

Kuis 1  
Struktur Data dan Algoritma  
Fakultas Ilmu Komputer Universitas Indonesia  
Durasi: 30 menit

**Petunjuk**: Tuliskan jawaban Anda sesingkat-singkatnya (**tanpa penjelasan apapun!**) di dalam kotak jawaban yang disediakan pada kolom paling kanan.

|  |  |  |
| --- | --- | --- |
| **No.** | **Soal** | **Jawaban** |
|  | Urutkan kompleksitas algoritma berikut ini, mulai dari yang paling rendah (paling kecil *growth rate*-nya) sampai yang paling tinggi (paling besar *growth rate*-nya).   * 1. O(N!)   2. O(2N (N/2)!)   3. O(N2 . log √N)   4. O(log(2N).log(10N))   5. O(log N3. log N2)   6. O(√N logN2) | **Tuliskan huruf-hurufnya saja:**  **e, f, d, c, b, a** |
|  | Algoritma A dan B kompleksitasnya sama yaitu , tentukan apakah pernyataan berikut ini BENAR atau SALAH: | **BENAR/SALAH:** |
| * + - * 1. Untuk data yang besar waktu eksekusi A pasti sama dengan B. | Salah |
| * + - * 1. Untuk data yang kecil waktu eksekusi A pasti sama dengan B. | Salah |
| * + - * 1. Jika untuk data berukuran N yang besar waktu eksekusi A dua kali waktu eksekusi B, maka untuk ukuran data 3N, waktu eksekusi A akan sama dengan waktu eksekusi B. | Salah |
| * + - * 1. Jika untuk data berukuran N yang besar waktu eksekusi A dua kali waktu eksekusi B, maka untuk ukuran data 3N, waktu eksekusi A akan sama dengan 6 kali waktu eksekusi B. | Salah |
|  | Apakah kompleksitas waktu potongan kode di bawah ini dalam notasi Big-Oh?  for (int i = 1; i <= N; i \*= 6) {  for (int j = 1; j <= i\*N; j += i) {  System.out.println(i+” ”+j);  }  } | O(N log N) |
|  | Diberikan fungsi rekursif berikut:  int f(int n) {  if (n <= 1) return n;  else if (n <= 5) return f(n-2)+n;  else return f(n/2)+n;  }  Berapakah nilai f(35)? | 66 |
|  | Diberikan fungsi rekursif algoritma binary search sebagai berikut:  int binsearch(data[ ], x, low, high){  mid = (low+high) / 2;  if( data[mid] == x )  return mid;  else if ( x < data[mid] )  return binsearch(data[ ], x, low, mid );  else  return binsearch(data[ ], x, mid+1, high);  }  Jika a adalah sebuah *sorted array* berisi bilangan 1 sampai dengan 100 (berturut-turut), berapa kali *recursive call* yang terjadi pada pemanggilan binsearch(a, 97, 0, a.length-1)termasuk pemanggilan pertama ? | 5 |
|  | Apa output dari potongan program di bawah ini?  String[] values = {"Sumatera", "Kalimantan", "Jawa", "Sulawesi", "Bali", "Papua"};  Deque<String> queueSatu = new ArrayDeque<String> ();  Stack<String> stackDua = new Stack<String> ();    for(int i = 0; i < values.length; i++) {  if(i % 2 == 0) {  queueSatu.add(values[i]);  }  else {  stackDua.push(values[i]);  }  }    while(!queueSatu.isEmpty()) {  String x = queueSatu.poll();  stackDua.push(x);  }    while(!stackDua.empty()) {  System.out.println(stackDua.pop());  } | Bali, Jawa, Sumatera, Papua, Solawesi, Kalimantan |
|  | Jika *swapping* adalah proses menukarkan isi dua elemen array, berapa kali dilakukan proses *swapping* oleh algoritma **Selection-sort** jika dijalankan untuk mengurutkan (dari kecil ke besar) array berikut ini:  {10, 9, 8, 7, 6, 5, 4, 3, 2, 1} ? |  |