|  |
| --- |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 1  1. (УСТНО) **Укажите для каждой пары функций имеет ли место соотношение** *fi(n)=O[fj(n)]* или *fi(n)=Ω[fj(n)] при n→∞*. а) *f1(n)=n2*;  б) *f2(n)=n2+100n*;  в) *f3(n)={n, если n – нечетно; n3, если n – четно};*  2. Реализовать при помощи одного массива два стека, суммарное количество элементов в которых ограничено длиной массива. Все стандартные команды над стеками должны выполняться за время, ограниченное константой. |
|  |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 2  1. (УСТНО) **Укажите для каждой пары функций имеет ли место соотношение** *fi(n)=O[fj(n)]* или *fi(n)=Ω[fj(n)] при n→∞*. а) *f1(n)=n2*;  б) *f2(n)=n2+100n*;  *в) f4(n)={n, если n≤100; n3, если n>100}.*  2. Предложить способ моделирования структуры данных *очередь*, используя два *стека*. Указание: «пристыковать» стеки своими вершинами, добавляя элементы в 1-й, а извлекая из второго; переваливая, если 2-й опустел, все из 1-го во 2-ой. |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 3  1. (УСТНО) **Найти функцию T(n) временной сложности алгоритмов (в худшем случае) и её асимптотический порядок роста: Перемножение квадратных матриц размера n.** *for i:=1 to n do  for j:=1 to n do  begin  C[i,j]:=0;  for k:=1 to n do  C[i,j]:=C[i,j]+ A[i,k]\*B[k,j]; end;*  2. Деком называют структуру данных, представляющую собой линейный список, в котором класть и забирать элементы можно с обеих сторон (начала и конца). Реализовать два варианта для программирования дека. 1) Использовать линейный список в динамической памяти. 2) Использовать массив фиксированной длины, индексы которого считать зацикленными по кругу, и две переменные: индекс начала дека и количество элементов. |
|  |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 4  1. (УСТНО) **Найти функцию T(n) временной сложности алгоритмов (в худшем случае) и её асимптотический порядок роста.**  ***procedure mystery(n:integer);*** *Var i,j,k: integer;  begin  for i:=1 to n-1 do  for j:=i+1 to n do  for k:=1 to j do {Группа операторов со временем O(1)}  end;*  2. Напечатать в порядке возрастания первые n натуральных чисел, в разложение которых на простые множители входят только числа 2, 3, 5.  Указание: ввести три очереди: x2, x3, x5, в которых будут хранится ещё не напечатанные числа, которые соответственно в 2, 3, 5 раз больше, чем напечатанные. Печатая число x, помещаем числа 2x, 3x, 5x в соответствующие очереди. Для печати выбираем наименьшее из чисел, стоящих вначале очередей x2, x3, x5. Начать с печати числа 1. |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 5  1. (УСТНО) **Найти функцию T(n) временной сложности алгоритмов (в худшем случае) и её асимптотический порядок роста.**  ***procedure veryodd(n:integer);*** *Var i,j,x.y :integer;  begin  for i:=1to n do  if нечет(i) then  begin  for j:=1 to n do x:=x+1;  for j:=1 to I do y:=y+1;  end;  end;*  2. Известно, что ориентированный граф связен, то есть из любой вершины можно пройти в любую по направленным ребрам. Кроме того, из каждой вершины выходит столько же ребер, сколько входит. Доказать, что существует замкнутый цикл, проходящий по каждому ребру ровно один раз. Составить алгоритм нахождения такого цикла. Указание. Использовать дек для текущего хранения «змеи» - пути по направленным ребрам (от «хвоста» к «голове»). Начиная с произвольной вершины, пока змея не включает в себя все ребра графа, делаем следующее. Если из головы выходит хотя бы одно не использованное ребро, удлиняем змею. В противном случае – голова змеи в той же вершине что и хвост (доказать); тогда отрезать последнее звено «хвоста» и добавить его к «голове» («змея откусывает конец своего хвоста»). |
|  |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 6  1. (УСТНО) **Найти функцию T(n) временной сложности алгоритмов (в худшем случае) и её асимптотический порядок роста: сортировка методом пузырька.**  *procedure buble(Var A:array[1..n] of integer); Var i,j,t :integer; begin  for i:=1 to n-1 do  for j:=n downto i+1 do  if A[j-1]>A[j] then  begin  t:=A[j-1]; A[j-1]:=A[j]; A[j]:=t;  end; end;*  2. Реализовать структуру данных “***k*** очередей”, суммарная длина которых в любой момент не превышает ***n***, используя один массив ***A*** длины n и дополнительные переменные общим объемом не более С(n+k).  Указание. 1) Использовать массив ***Next[]*** длины ***n***, в котором ***Next[i]*** содержит указатель (номер в массиве ***A***) на следующий после ***i***-ого элемент в его очереди. 2) Для каждой ***j***-ой из ***k*** очередей хранить указатель ***First[j]*** (номер в массиве ***A***) начального элемента (и по желанию конечного). 3) Все свободные ячейки массива ***A*** также объединить в очередь с начальным элементом ***Free***. |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 7  1. (УСТНО) **Найти функцию T(n) временной сложности алгоритмов (в худшем случае) и её асимптотический порядок роста: ф*акториал***  *function factorial(n:integer):integer; begin  if n≤1 then factorial:=1  else factorial:=n\*factorial(n-1); end;*  2. Реализовать структуру данных “***k*** деков”, суммарная длина которых в любой момент не превышает ***n***, используя один массив ***A*** длины n и дополнительные переменные общим объемом не более С(n+k).  Указание. 1) Использовать массив ***Next[]*** длины ***n***, в котором ***Next[i]*** содержит указатель (номер в массиве ***A***) на следующий после ***i***-ого элемент в его очереди. 2) Для каждой ***j***-ой из ***k*** очередей хранить указатель ***First[j]*** (номер в массиве ***A***) начального элемента (и по желанию конечного). 3) Все свободные ячейки массива ***A*** также объединить в очередь с начальным элементом ***Free***. |
|  |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 8  1. (УСТНО) **Найти функцию T(n) временной сложности алгоритмов (в худшем случае) и её асимптотический порядок роста:**  ***Числа Фибоначчи***  *function fib(n:integer):integer; begin  if n≤1 then fib:=1   else fib:=fib(n-2)+fib(n-1); end;*  2. Задан массив вещественных чисел x0, x1, ...,xn-1. Произвести сортировку массива по возрастанию методом "***последовательного выбора минимума"***. (То есть переставить значения элементов массива так, чтобы выполнялись неравенства x0≤ x1,≤ ...,≤xn-1.) |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 9  1. (УСТНО) **Докажите, что следующие утверждения истинны:** а) 17 имеет порядок O(1); б) n(n-1)/2 имеет порядок O(n2) при n→∞;  2. Задан массив вещественных чисел x0, x1, ...,xn-1. Произвести сортировку массива по возрастанию "***пузырьковым*** методом". (То есть переставить значения элементов массива так, чтобы выполнялись неравенства x0≤ x1,≤ ...,≤xn-1.) |
|  |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 10  1. (УСТНО) **Докажите, что следующие утверждения истинны:**  max{n3, 10n2}=O(n3) при n→∞;  2. Задан массив вещественных чисел x0, x1, ...,xn-1. Произвести сортировку массива по возрастанию методом "***вставки с линейным поиском места вставки"***. (То есть переставить значения элементов массива так, чтобы выполнялись неравенства x0≤ x1,≤ ...,≤xn-1.) |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 11  1. (УСТНО) **Докажите, что следующие утверждения истинны:**  г) 1k+2k+…nk=O(nk+1), Ω(nk+1) при n→∞;  2. Задан массив вещественных чисел x0, x1, ...,xn-1. Произвести сортировку массива по возрастанию методом "***вставки с бинарным поиском места вставки"***. (То есть переставить значения элементов массива так, чтобы выполнялись неравенства x0≤ x1,≤ ...,≤xn-1.) |
|  |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 12  1. (УСТНО) **Докажите, что следующие утверждения истинны:**  д) пусть p(n) – полином степени k (k>0) p(n)=aknk+…+a1n+a0, ak>0. Докажите, что p(n)=O\*(nk) и, следовательно, p(n)=O(nk), p(n)=Ω(nk) при n→∞.  2. Задан массив вещественных чисел x0, x1, ...,xn-1. Произвести сортировку массива по возрастанию методом "***файловой сортировки слияниями***". (То есть переставить значения элементов массива так, чтобы выполнялись неравенства x0≤ x1,≤ ...,≤xn-1.) |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 13  1. (УСТНО) **Расположите следующие функции в порядке степени роста:** ; ; ; ; ; ; ; ; ; 17.  2. Задан массив вещественных чисел x0, x1, ...,xn-1. Произвести сортировку массива по возрастанию методом "***бинарного дерева***". (То есть переставить значения элементов массива так, чтобы выполнялись неравенства x0≤ x1,≤ ...,≤xn-1.) |
|  |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 14  1. (УСТНО) Пусть сортировка массива (или ряда камней) производится при помощи: 1) функции ***тяжелее(I,j:integer):boolean****;* 2) перестановки двух произвольных элементов. Доказать, что временная сложность всякого такого алгоритма A удовлетворяет неравенству T(n)≥log2n! Вывести отсюда, что T(n) ≥ 1/2 ∙n log2n. **Указание**: 1) Пусть на вход подаётся расположение из n камней. Имеется всего n! таких расположений. 2) Каждому расположению поставим в соответствие протокол взвешиваний при работе алгоритма A: b1b2…bd (bi∈{0,1} результаты функции ***тяжелее***, d – максимальное количество взвешиваний, потребовавшееся для всех расположений). 3) Протоколы b1b2…bd для всех n! расположений различны. Сравнить с количеством бинарных строк длины d.  2. Задан массив вещественных чисел x0, x1, ...,xn-1. Произвести сортировку массива по возрастанию методом "***быстрой сортировки***" ***по*** ***Хоару***. (То есть переставить значения элементов массива так, чтобы выполнялись неравенства x0≤ x1,≤ ...,≤xn-1.) |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 15  1. (УСТНО) 1) Доказать, что задача нахождения выпуклой оболочки n точек на плоскости сводится к задаче сортировки n чисел. То есть, может быть решена за время не большее, чем С∙n∙log n.  2. Имеется массив целых чисел a[1],…a[n], принимающих значения {0,1,…,m}. Сортировать этот массив по возрастанию, используя количество действий порядка (m+n). Указание: Посчитать сколько раз каждое из чисел {0,1,…,m} встречается в массиве. |
|  |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 16  1. (УСТНО) 2) Доказать, что задача сортировки n чисел сводится к задаче нахождения выпуклой оболочки n точек на плоскости. То есть, в общем случае не может быть решена быстрее, чем за время С∙n∙log n.  2. В массиве a[1],…a[n] целых чисел переставить элементы так, чтобы четные числа шли перед нечетными (не меняя взаимный порядок внутри каждой из групп) за время C∙n. |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 17  1. (УСТНО) **Укажите для каждой пары функций имеет ли место соотношение** *fi(n)=O[fj(n)]* или *fi(n)=Ω[fj(n)] при n→∞*. а) *f1(n)=n2*;  б) *f2(n)=n2+100n*;  в) *f3(n)={n, если n – нечетно; n3, если n – четно};*  2. Имеется массив x[1], …,x[n] из целых чисел, принимающих значения 0..2k-1, каждое из которых удобно рассматривать как k-битовое двоичное число: *x=bk-1bk…b1b0, bi∈{0,1}*. Используя доступ к *i*-тому биту при помощи функции Bit(x,i) и сравнения вида Bit(x,i)=0, Bit(x,i)=1, отсортировать массив X[] за время n∙k. (Указание: сначала сортировать массив по 0-му биту, затем по 1-му и т.д. до (k-1)-ого.) |
|  |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 18  1. (УСТНО) **Укажите для каждой пары функций имеет ли место соотношение** *fi(n)=O[fj(n)]* или *fi(n)=Ω[fj(n)] при n→∞*. а) *f1(n)=n2*;  б) *f2(n)=n2+100n*;  *в) f4(n)={n, если n≤100; n3, если n>100}.*  2. Даны n чисел x[1], …,x[n] и функция f(x), принимающая на них значения 1,2,…,m. Переставить числа так, чтобы значения функции f(x) на них не убывали (сохраняя порядок чисел с равными значениями).  Число действий порядка m+n (а не m∙n ).  *Указания* (не обязательные для исполнения). Можно воспользоваться идеями двух вариантов выполнения задачи. 1) Использовать m очередей (организованных в динамической памяти либо в одном массиве длины n). 2) На первом проходе вычислить количество членов массива принимающих каждое из m значений функции f. Разделив рабочий массив на отрезки соответствующей длины, заполнить его за 2-ой проход. |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 19  1. (УСТНО) **Найти функцию T(n) временной сложности алгоритмов (в худшем случае) и её асимптотический порядок роста: Перемножение квадратных матриц размера n.** *for i:=1 to n do  for j:=1 to n do  begin  C[i,j]:=0;  for k:=1 to n do  C[i,j]:=C[i,j]+ A[i,k]\*B[k,j]; end;*  2. Дано n чисел в диапазоне 0…p2-1. Как сортировать их, сделав порядка n действий?  Указание (не обязательное). Числа x∈{0,1,…,p2-1} можно представить в p-ичной записи: x=b1∙p+b0, где b1, b0 ∈ {0,…,p-1}. За первый проход (или два прохода) можно отсортировать их по значениям b0, за второй по b1 (не считая затрат времени на копирование). |
|  |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 20  1. (УСТНО) **Найти функцию T(n) временной сложности алгоритмов (в худшем случае) и её асимптотический порядок роста.**  ***procedure mystery(n:integer);*** *Var i,j,k: integer;  begin  for i:=1 to n-1 do  for j:=i+1 to n do  for k:=1 to j do {Группа операторов со временем O(1)}  end;*  2. Найти количество различных чисел среди элементов данного массива длины n. Число действий порядка n∙log n. |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 21  1. (УСТНО) **Найти функцию T(n) временной сложности алгоритмов (в худшем случае) и её асимптотический порядок роста.**  ***procedure veryodd(n:integer);*** *Var i,j,x.y :integer;  begin  for i:=1to n do  if нечет(i) then  begin  for j:=1 to n do x:=x+1;  for j:=1 to I do y:=y+1;  end;  end;*  2. Дано n отрезков: [ai, bi], i=1,…,n. Найти максимальное число ***k***, для которого существует точка прямой, покрытая ***k*** отрезками заданного набора. Число действий порядка n∙log n. |
|  |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 22  1. (УСТНО) **Найти функцию T(n) временной сложности алгоритмов (в худшем случае) и её асимптотический порядок роста: сортировка методом пузырька.**  *procedure buble(Var A:array[1..n] of integer); Var i,j,t :integer; begin  for i:=1 to n-1 do  for j:=n downto i+1 do  if A[j-1]>A[j] then  begin  t:=A[j-1]; A[j-1]:=A[j]; A[j]:=t;  end; end;*  2. Дано n точек на плоскости, заданных своими декартовыми координатами. За время n∙log n построить (n-1)-звенную не пересекающую себя ломаную, проходящую через все точки. Указание: использовать сортировку. |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 23  1. (УСТНО) **Найти функцию T(n) временной сложности алгоритмов (в худшем случае) и её асимптотический порядок роста: ф*акториал***  *function factorial(n:integer):integer; begin  if n≤1 then factorial:=1  else factorial:=n\*factorial(n-1); end;*  2. Дано n точек на плоскости, заданных своими декартовыми координатами. За время n∙log n построить n-звенную замкнутую не пересекающую себя ломаную, проходящую через все точки. Указание: использовать сортировку. |
|  |
| Комбинаторные алгоритмы (III курс)  Контрольная работа № 1.  Временная сложность алгоритмов. Структуры данных. Сортировка.  Вариант 24  1. (УСТНО) **Найти функцию T(n) временной сложности алгоритмов (в худшем случае) и её асимптотический порядок роста:**  ***Числа Фибоначчи***  *function fib(n:integer):integer; begin  if n≤1 then fib:=1   else fib:=fib(n-2)+fib(n-1); end;*  2. Дано n точек на плоскости, заданные своими декартовыми координатами. За время n∙log n построить их выпуклую оболочку (то есть выпуклый многоугольник с вершинами в точках множества, содержащий все точки этого множества). Указание (не обязательное): упорядочить точки по углам лучей, проведенных из центра масс ко всем точкам множества. |