Лекция 5.

1.Можно ли решить NP-полные задачи за полиномиальное время?

Напомню, что любую задачу из класса NP можно «полиномиально свести» к любой из NP-полных задач. Значит если если существует полиномиальный алгоритм для решения хотя бы одной задачи, то потенциально любую NP-полную задачу можно также решить полиномиальным алгоритмом.

На данный момент это невозможно.

2. Можно ли решить задачи класса Р за полиномиальное время?

Задачи класса P – это интуитивно задачи, решаемые за реальное время.

3. Можно ли решить задачи класса NР за полиномиальное время?

Поэтому некоторые задачи NP класса можно решить при помощи недетерминированного алгоритма в полиномиальное время.

4.Можно ли найти приближенное решение для задачи класса NР (хотя бы для одной задачи из этого класса) за полиномиальное время?

Да, см пункт.

5. Можно ли найти приближенное решение для NР-трудной задачи за полиномиальное время (хотя бы для какой-нибудь одной задачи из этого класса)?

С помощью алгоритмов, в которых кол-во выполняемых операций растёт по экспонециальному закону, можно решить лишь задачи очень малых размеров.

6. Можно ли найти приближенное решение для NР-полной задачи за полиномиальное время (хотя бы для какой-нибудь одной задачи из этого класса)?

1. Задача должна принадлежать классу NP (L ∈ NP) 2. К ней полиномиально должны сводиться все задачи из класса NP (Lx ≤P L, для каждого Lx ∈ NP)

Да.

7. Существуют ли задачи, которые не могут быть решены никаким алгоритмом?

Некоторые задачи вообще не могут быть решены никаким алгоритмом. Классический пример такой задачи — «проблема остановки» (выяснить останавливается ли данная программа на данном входе).

8. Существуют задачи, для которых существует решающий их алгоритм, но любой такой алгоритм работает долго – время его работы не есть О(*nk*) ни для какого фиксированного числа k?

Кроме того, бывают задачи, для которых существует решающий их алгоритм, но любой такой алгоритм работает «долго» — время его работы не есть O(nk) ни для какого фиксированного числа k.

9. В чем состоит в теории алгоритмов проблема, называемая «Р ≠ NP»?

Если существует задача, принадлежащая классу NPC, для которой существует полиномиальный алгоритм решения F = O(n k ), то класс P совпадает с классом NP, т.е. P = NP .

После введения в теорию алгоритмов понятий сложностных классов Эдмондсом (Edmonds, 1965) была поставлена основная проблема теории сложности: P =? NP которая до сих пор не решена!

Словесная формулировка проблемы имеет вид: можно ли все задачи, решение которых проверяется с полиномиальной сложностью, решить за полиномиальное время?

Очевидно, что любая задача, принадлежащая классу P, принадлежит и классу NP, т.к. она может быть полиномиально проверена – задача проверки решения может состоять просто в повторном решении задачи.

10. В чем состоит *интересное* свойство NP-полного класса задач?

Класс NP- полных задач обладает следующими свойствами.

1. Никакую NP-полную задачу нельзя решить никаким известным полиномиальным алгоритмом, несмотря на настойчивые усилия многих блестящих исследователей.

2. Если бы удалось построить полиномиальный алгоритм для какой-нибудь NP-полной задачи, то это бы означало полиномиальную разрешимость каждой NP-полной задачи.

3. Задача должна принадлежать классу NP (L ∈ NP)

4. К ней полиномиально должны сводиться все задачи из класса NP (Lx ≤P L, для каждого Lx ∈ NP)

5.Если существует задача, принадлежащая классу NPC, для которой существует полиномиальный алгоритм решения F = O(n k ), то класс P совпадает с классом NP, т.е. P = NP .

11. Что означает понятие – «***Класс задач Р, состоящий из «задач» разрешимых за полиномиальное время***»

Про задачу П говорят, что она разрешима за полиномиальное время, если существует машина Тьюринга Т, решающая ее за полиномиальное время. Обозначим через Р класс задач, разрешимых за полиномиальное время.

Задача называется полиномиальной, т.е. относится к классу P, если существует константа k и алгоритм а, решающий задачу с трудоемкостью

Fa(n) = O( n k ),

где n - длина входных данных алгоритма в битах n = |D|. Задачи класса P – это интуитивно задачи, решаемые за реальное время.

12. Укажите свойства P класса задач?

Преимущества алгоритмов из класса Р:

• для большинства задач из класса P константа k < 6

• класс P обладает свойством естественной замкнутости (сумма или произведение полиномов есть полином)

• класс P инвариантен по модели вычислений (для широкого класса моделей)

13. Задача (проблема, язык) *L*  {0, 1}\* называется **NP-полной** (NPC), если

Определение класса NPC (NP-complete) или класса NP-полных задач требует выполнения следующих условий:

1. Задача должна принадлежать классу NP (L ∈ NP)

2. К ней полиномиально должны сводиться все задачи из класса NP (Lx ≤P L, для каждого Lx ∈ NP)

<https://infopedia.su/3x45b1.html>

14. Задача (проблема, язык) *L*  {0, 1}\* называется **NP-трудной** (NPН), если

Хз

15. Укажите основное свойство NP-полных задач

Класс NP- полных задач обладает следующими свойствами.

1. Никакую NP-полную задачу нельзя решить никаким известным полиномиальным алгоритмом, несмотря на настойчивые усилия многих блестящих исследователей.

2. Если бы удалось построить полиномиальный алгоритм для какой-нибудь NP-полной задачи, то это бы означало полиномиальную разрешимость каждой NP-полной задачи.

полиномиальная сводимость

16. Если для задачи (проблемы, языка) L найдется задача (проблема, язык) L' *∈* NPC, для которого L' *<=p* L, то задача L <…> (*Окончите фразу*)*.*

Принято говорить, что задача задается некоторым языком, тогда если задача 1 задана языком L1, а задача 2 – языком L2, то полиномиальная сводимость обозначается следующим образом: L2 =< pL1.

Определение класса NPC (NP-complete) или класса NP-полных задач требует выполнения следующих двух условий:

во-первых, задача должна принадлежать классу NP (L є NP), и,

во-вторых, к ней полиномиально должны сводиться все задачи из класса NP (Lx=< pL, для каждого Lx є NP)

17. Всякий ли NP-полный язык является NP-трудным?

Задача является NP-трудной (или NP-сложной), если каждая задача из класса NP полиномиальносводится к ней. Задача является NP-полной,если она входит в класс NP и является NP-трудной.(ответ Да)

18. Всякий ли NP-трудный язык является NP-полным?

Задача является NP-трудной (или NP-сложной), если каждая задача из класса NP полиномиальносводится к ней. Задача является NP-полной,если она входит в класс NP и является NP-трудной.(ответ нет)

19. Что требуется, чтобы NP-трудная задача являлась и NP-полной?

Задача называется NP-трудной если каждая задача из NP полиномиально сводится к ней. NP-трудная задача имеет тот смысл, что эта задача не проще, чем «самая трудная в NP». В классе NP выделяются NP-полные задачи. Задача называется NP- полной, если она входит в NP и каждая задача из NP полиномиально сводится к ней (т.е. является NP-трудной). NP-полные задачи понимаются как самые трудные задачи из класса NP.

**Тема 2 Асимптотическая оценка трудоемкости кода программы**

**Лекция 6**

**1.** Определите асимптотическую оценку O(*f*(*N*)) функции роста трудоемкости следующего фрагмента программы: int F(A[][][],n) {int i, j, k, s=0; for(i=1; i<=n; i+=2) for(j=1; j<=n-1; j\*=3) for(k=1; k<=n-2; k++) s=s+A[i][j][k]; return s;}

**N^3**

**Тема 3 Рекурсия**

**1**Чему равен ***порядок*** рекуррентного соотношения

Порядок равен 2.

**2.** Укажите формулу, являющуюся ***решением* (**ПРИМЕРОМ решения**)** некоторого рекуррентного уравнения:

?

?

?

F(n) = F(n-1) + 10

**3**Решение рекуррентного уравнения может быть

Метод итераций, метод рекурсивных деревьев, подстановочный метод

4. В ходе метода прямой подстановки выполняются следующие вычисления

Начиная с начального члена (или членов) последовательности для данных начальных условий, мы можем использовать рекуррентное соотношение для генерации нескольких первых членов его решения в надежде понять, как именно может выглядеть конечная формула. Если такая формула найдена, ее корректность должна быть проверена подстановкой в рекуррентное уравнение и в начальные условия или доказана методом математической индукции. С практической точки зрения этот метод работает только для очень ограниченного количества случаев, поскольку обычно очень сложно распознать вид формулы по нескольким первым членам последовательности.

5. В ходе метода обратной подстановки выполняются следующие вычисления

Этот метод решения рекуррентных соотношений работает так, как указано в его названии: используя рассматриваемое рекуррентное соотношение, мы выражаем x(n-1) как функцию от x(n-2). Повторение этого шага для x(n-2) дает выражение x(n) как функцию от х(n-i) для произвольного i = 1, 2, 3… Выбирая i так, чтобы n-i достигало начального условия и используя одну из формул суммирования, зачастую удается получить явную формулу для решения рекуррентного соотношения.

6. В ходе метода обратной подстановки выполняются следующие вычисления

?

?

F(n) = F(n-(i-1) + (i-1)

7. Данное дерево рекурсии может соответствовать рекуррентному выражению: 

–> +

–>

**Тема 4 Асимптотика вложенности сумм**

–> , *k=const +*

–>  +

–>  +

–>  -

**Тема 5 Соответствие асимптотической оценки трудоемкости кода программы**

Определите фрагмент программы, соответствующий асимптотической оценке O(*f*(*N*))= n3:

int F(A[][][],n) {int i, j, k, s=0; for(i=1; i<=n\*n; i++) for(j=1; j<=n-1; j++);}

**Тема 6. Основные понятия Анализа сложности алгоритма и программы**

1. Величина, отражающая порядок величины требуемого ресурса (времени или дополнительной памяти) в зависимости от размерности задачи – это … (закончите определение)

Сложность алгоритмов

2. Сложность алгоритма бывает … (какой термин может быть применим)

1 временная T(n)

2Пространственная(ёмкостная) V(n)

3. На время выполнения программы влияет

1 зависит от временной сложности алгоритма

2 исходные данные d(n), от их “размера”

3 F(N)- Время выполнения алгоритма.

3 Время выполнения программы измеряется

Физические единицы измерения времени(секунды, миллисекунды, ”Тики”) ЦП

**Тема 7. Основные понятия аппарата О-символики**

1. Обозначение O(*g*(*n*) – это

асимптотически точная оценка функции роста трудоемкости алгоритма

1. Про запись *f*(*n*) = O(*g*(*n*)) можно сказать,

(\* *f*(*n*) является представителем множества всех функций, порядок роста которых при достаточно больших *n* )

Ответ : что g(n) является асимптотически точной оценкой для f(n)

3. Если для функции *T*(*n*) найдутся константы *c*1 > 0 и *c*2 > 0 и такое число *n*0 > 0, что выполнится условие *c*1*n*2 ≤ *T*(*n*) ≤ *c*2*n*2, то

Говорят, что время выполнения алгоритма (программы) асимптотически точно соответствует функции n^2

T(n)=Θ(n^2)

4. Если для функций *X*(*n*) >0(==f(n)) и *Y*(*n*) >0(==g(n)) найдутся константы *c*1 > 0 и *c*2 > 0 и такое число *n*0 > 0, что выполнится условие *c*1*Y(n)*≤ *X*(*n*) ≤ *c*2 *Y(n)*, то

Говорят

C1(g(n))<=f(n)<=C2((g(n))

X(n)=Θ(Y(n))

5. Если для функций *X*(*n*) >0 и *Y*(*n*) >0 найдутся константы *c*1 > 0 и *c*2 > 0 и такое число *n*0 > 0, что выполнится условие

C1(Y(n))<X(n)<C2((Y(n))

6. Если для функций *X*(*n*) >0 и *Y*(*n*) >0 найдется константа *c*1 > 0 и такое число *n*0 > 0, что выполнится условие 0 < *c*1*Y(n)*≤ *X*(*n*), то

X(n)=Ω(Y(n))

7. Если для функций *X*(*n*) >0 и *Y*(*n*) >0 найдется константа *c*1 > 0 и такое число *n*0 > 0, что выполнится условие 0≤ *X*(*n*) ≤ *c*1*Y*(*n*), то

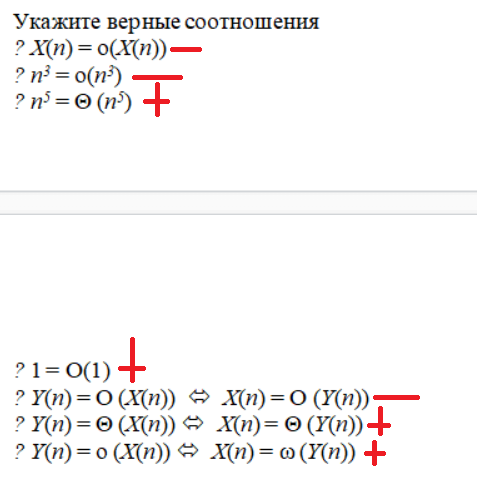
X(n)=O(Y(n))

Для номеров 8,9,10,11,12

Используйте табличку:

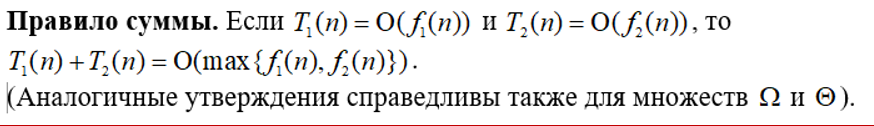
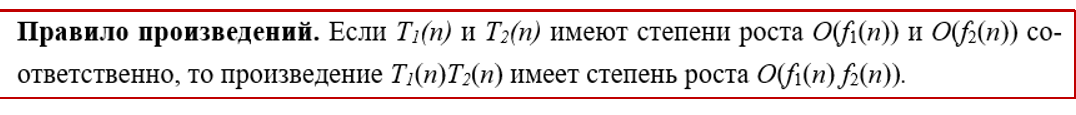


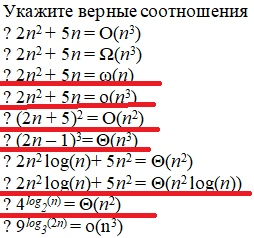
**Тема 8. Свойства О-символики (транзитивность, рефлексивность, симметричность, обратимость)**



+ таблица из 7 темы

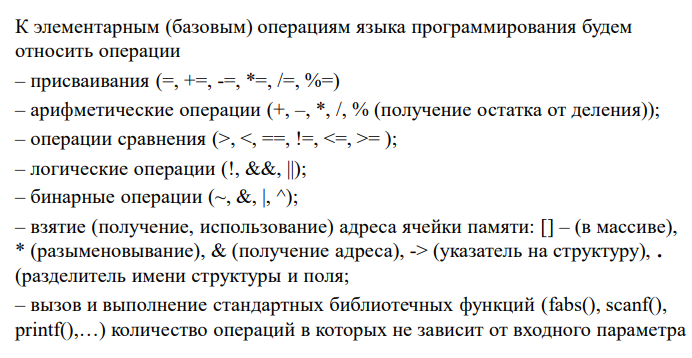
**Тема 9** **Сложение и умножение в O-символике**

1. N^2
2. 
3. 



**Тема 11 Принципы анализа алгоритмов и программ: базовые операции**

1. К базовым (элементарным) операциям, трудоемкость которых определяется как O(1), относятся:



**Тема 12 Принципы анализа алгоритмов и программ: выполнение операторов**

**1** Трудоемкость выполнения линейной последовательности операторов определяется

обычно имеет порядок О(1), как всякий линейный оператор, кроме случаев использования внутри них функций и процедур. В этом случае оператор имеет порядок выполнения функции или процедуры, если возможно тем или иным образом оценить трудоемкость подпрограммы.

2. Трудоемкость вычисления выражения (в круглых скобках) определяется

Трудоемкость вычисления выражения (в круглых скобках) определяется с помощью правила сумм. Поэтому степень роста времени выполнения последовательности операторов без определения констант пропорциональности совпадает с наибольшим временем выполнения оператора в данной последовательности.

3. Трудоемкость выполнения оператора условия if определяется

Трудоемкость выполнения оператора условия if определяется так: если конкретную ветвь, которая будет выполнена, определить затруднительно, то из всех возможных исходов выбирается самый «трудоёмкий», тем самым определяется время работы в худшем, т.е. O(f(N)) : • в оператора if – их два: ветвь «да» и ветвь «нет», трудоемкость каждой определяем отдельно, а затем выбираем ту ветвь, функция роста трудоемкости которой имеет бОльший порядок;

**4.** Трудоемкость выполнения оператора выбора switch определяется

• в операторе выбора switch анализируется каждая case ветвь, включая ветвь default трудоемкость каждой определяем отдельно, а затем выбираем ту ветвь, функция роста трудоемкости которой имеет бОльший порядок.

5. Укажите факторы, которые надо учитывать при определении трудоемкости выполнения оператора цикла for(;;)

Часто время выполнения цикла вычисляется, пренебрегая определением констант пропорциональности, как произведение количества выполненных итераций цикла на наибольшее возможное время выполнения операторов тела цикла. Однако, не следует для циклов перемножать число выполняемых итераций на трудоемкость отдельной итерации, так как в этом случае зачастую не верно определяется количество исполняемых операций (вид функции роста), хотя асимптотическая оценка может оказаться верной. Кроме того, время выполнения каждого цикла, если в программе их несколько, должно определяться отдельно. В случае вложенных циклов, анализ надо начинать оценивания трудоемкость тела цикла «самого внутреннего» цикла. Тогда полученная для него трудоемкость будет использоваться для оценки трудоемкости тела цикла, внешнего относительно рассматриваемого цикла. Так последовательно от внутреннего к самому внешнему циклу можно построить функцию роста. Поскольку каждый из циклов характеризуется суммированием то для вложенных циклов характерна функция роста вида содержащая несколько вложенных сумм, которые затем необходимо последовательно раскрыть, начиная с самого вложенного.

6. Трудоемкость цикла for(;;) вычисляется

Время выполнения цикла является суммой времени всех исполняемых итераций цикла, в свою очередь состоящих из времени выполнения операторов тела цикла и времени вычисления условия прекращения цикла (обычно последнее имеет порядок О(1), если не содержит внутри себя функцию от n).