ПЕРМСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ

ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Фонды оценочных средств по дисциплине «Высокоэффективные алгоритмы»

Направление подготовки 01.04.02 «Прикладная математика и информатика»

1. Формируемые дисциплиной компетенции

ОПК 2.3. Реализует математический метод на языке программирования высокого уровня и/или с помощью специализированных пакетов программ

ОПК 4.1. Комбинирует и адаптирует современные информационно-коммуникационные технологии для реализации решения математических задач

2. Планируемые результаты обучения

Коды компетенций	**Планируемый результат**
ОПК 2.3	ЗНАТЬ: основные термины, характеристики и подходы к созданию параллельных алгоритмов; постановки классических задач, синтаксис нескольких парадигм параллельного программирования. УМЕТЬ: запрограммировать выбранный алгоритм на языке программирования с применением технологии MPI, сформировать отчёт по производительности алгоритма и программы. ВЛАДЕТЬ: навыками тестирования программы на правильность и эффективность выполнения.
ОПК 4.1	ЗНАТЬ: основные вычислительные алгоритмы, используемые в высокоэффективных вычислениях, границы их применимости. УМЕТЬ: приводить обоснование по использованию той или иной парадигмы, алгоритма. ВЛАДЕТЬ: методами анализа эффективности параллельных алгоритмов

3. Спецификация теста

Тест по дисциплине «Высокоэффективные алгоритмы» состоит из 10 заданий. Рекомендованное время решения теста испытуемым — 30 минут. Верно решенное задание оценивается в 2 балла, максимальный балл за верное выполнение всех заданий теста — 20 баллов. Минимальный проходной балл — 9, что соответствует минимальному порогу для выставления отметки «удовлетворительно».

Схема конвертации баллов в отметки:

0-8 баллов — «неудовлетворительно»

9-12 баллов — «удовлетворительно»

13-16 баллов — «хорошо»

17-20 баллов — «отлично»

Структура теста:

Наименование раздела/темы	Планируемый результат	Количество заданий в тесте
Оценка сложности параллельного алгоритма	ЗНАТЬ: основные термины, характеристики и подходы к созданию параллельных алгоритмов; постановки классических задач, синтаксис нескольких парадигм параллельного программирования. УМЕТЬ: запрограммировать выбранный алгоритм на языке программирования с применением технологии MPI, сформировать отчёт по производительности алгоритма и программы. ВЛАДЕТЬ: навыками тестирования программы на правильность и эффективность выполнения.	4
Методы и теоремы анализа алгоритмов	ЗНАТЬ: основные термины, характеристики и подходы к созданию параллельных алгоритмов; постановки классических задач, синтаксис нескольких парадигм параллельного программирования. УМЕТЬ: запрограммировать выбранный алгоритм на языке программирования с применением технологии MPI, сформировать отчёт по производительности алгоритма и программы. ВЛАДЕТЬ: навыками тестирования программы на правильность и эффективность выполнения.	2
Решение рекуррентных уравнений	ЗНАТЬ: основные вычислительные алгоритмы, используемые в высокоэффективных вычислениях, границы их применимости. УМЕТЬ: приводить обоснование по	2

	использованию той или иной парадигмы, алгоритма. ВЛАДЕТЬ: методами анализа эффективности параллельных алгоритмов	
Параллельная сортировка	ЗНАТЬ: основные вычислительные алгоритмы, используемые в высокоэффективных вычислениях, границы их применимости. УМЕТЬ: приводить обоснование по использованию той или иной парадигмы, алгоритма. ВЛАДЕТЬ: методами анализа эффективности параллельных алгоритмов	1
Перемножение матриц	ЗНАТЬ: основные вычислительные алгоритмы, используемые в высокоэффективных вычислениях, границы их применимости. УМЕТЬ: приводить обоснование по использованию той или иной парадигмы, алгоритма. ВЛАДЕТЬ: методами анализа эффективности параллельных алгоритмов	1

Тест по дисциплине «Высокоэффективные алгоритмы». Вариант 1

1. Временная сложность алгоритма двоичного поиска в упорядоченном массиве

а. $O(n)$

б. $O(n^2)$

в. $O(\log_2 n)$

г. $O(\ln n)$

2. Рассмотрим последовательный алгоритм чётно-нечётной сортировки (см. псевдокод). Применим его к массиву из чисел $27, 1, 4, 19, 7, 10, 6, 2$. Как выглядит массив после итерации $i = 3$?

```
ODD-EVEN
begin
 for i=1 to n do begin
  if i mod 2 = 1 then
   for j=0 to n/2-1 do
    compare-exchange(b[2j+1],b[2j+2])
   if i mod 2 = 0 then
    for j=1 to n/2-1 do
     compare-exchange(b[2j],b[2j+1])
 end for
end ODD-EVEN
```

а. $27, 1, 4, 7, 19, 6, 10, 2$

б. $1, 4, 27, 7, 19, 2, 10, 6$

в. $1, 27, 4, 19, 7, 10, 2, 6$

г. $1, 4, 7, 27, 2, 19, 6, 10$

3. Как обозначается функция, которая асимптотически имеет тот же порядок, что и $f(x)$?

а. $O(f(x))$

б. $\Theta(f(x))$

в. $o(f(x))$

г. $f(x)$

4. Вычислительная сложность алгоритма решения Ханойской башни с n дисками

а. 2^n

б. n

в. $\log_2 n$

г. $2n$

5. Вычислительная сложность алгоритма перемножения матриц $n \times n$ по определению

а. $\Theta(n^2)$

б. $\Theta(n^3)$

в. $\Theta(n^{7/4})$

г. $\Theta(n \log n)$

6. Метод Карацубы используется для
   а. перемножения квадратных матриц
   б. нахождения корня произвольной функции
   в. перемножения многозначных чисел
   г. оценки вычислительной сложности алгоритмов

7. Основная теорема о рекуррентных соотношениях используется для
   а. решения линейных рекуррентных уравнений
   б. нахождения корня произвольной функции
   в. перемножения многозначных чисел
   г. оценки вычислительной сложности рекурсивных алгоритмов

8. Рекуррентное уравнение для нахождения вычислительной сложности обхода бинарного дерева записывается так
   а. $T(n) = T\left(\frac{n}{2}\right) + O(1)$
   б. $T(n) = 2T\left(\frac{n}{2}\right) + O(1)$
   в. $T(n) = 7T\left(\frac{n}{2}\right) + O(n^2)$
   г. $T(n) = 2T\left(\frac{n}{2}\right) + O(\log n)$

9. Рекуррентное уравнение для нахождения вычислительной сложности сортировки слиянием записывается так
   а. $T(n) = T\left(\frac{n}{2}\right) + O(1)$
   б. $T(n) = 2T\left(\frac{n}{2}\right) + O(1)$
   в. $T(n) = 7T\left(\frac{n}{2}\right) + O(n^2)$
   г. $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$

10. Функция изоэффективности $W = K(pT_p - W)$ для параллельного алгоритма определяет

а. условие для возникновения сверхлинейного ускорения

б. зависимость размера решаемой задачи от числа используемых процессоров для обеспечения постоянного уровня эффективности параллельных вычислений

в. зависимость эффективности от числа процессоров

г. оценку вычислительной сложности рекурсивных алгоритмов

Правильные ответы, вариант 1

Вар. ответа	Номер вопроса									
	1	2	3	4	5	6	7	8	9	10
а)				Х						
б)			Х		Х			Х		Х
в)	Х					Х				
г)		Х					Х		Х	

Тест по дисциплине «Высокоэффективные алгоритмы». Вариант 2

1. Временная сложность алгоритма двоичного поиска в неупорядоченном массиве

   а. $O(n)$

   б. $O(n^2)$

   в. $O(\log_2 n)$

   г. $O(\ln n)$

2. Рассмотрим последовательный алгоритм чётно-нечётной сортировки (см. псевдокод). Применим его к массиву из чисел $27,1,4,19,7,10,6,2$. Как выглядит массив после итерации $i = 2$?

```
ODD-EVEN
begin
 for i=1 to n do begin
   if i mod 2 = 1 then
 for j=0 to n/2-1 do
   compare-exchange(b[2j+1],b[2j+2])
 if i mod 2 = 0 then
   for j=1 to n/2-1 do
     compare-exchange(b[2j],b[2j+1])
 end for
end ODD-EVEN
```

   а. $27,1,4,7,19,6,10,2$

   б. $1,4,27,7,19,2,10,6$

   в. $1,27,4,19,7,10,2,6$

   г. $1,4,7,27,2,19,6,10$

3. Как обозначается функция, которая асимптотически растёт не быстрее, чем $f(x)$?

   а. $O(f(x))$

   б. $\Theta(f(x))$

   в. $o(f(x))$

   г. $< f(x)$

4. Сколько этапов требуется для вычисления суммы n чисел методом сдваивания?

   а. 2^n

   б. n

   в. $\log_2 n$

   г. 2

5. Вычислительная сложность алгоритма перемножения матриц $n \times n$ методом Штрассена

   а. $\Theta(n^2)$

   б. $\Theta(n^{\log_2 8})$

   в. $\Theta(n^{\log_2 7})$

г. $\Theta(n^3)$

6. Метод Карацубы используется для
   а. перемножения квадратных матриц
   б. нахождения корня произвольной функции
   в. перемножения многозначных чисел
   г. оценки вычислительной сложности алгоритмов

7. Основная теорема о рекуррентных соотношениях используется для
   а. решения линейных рекуррентных уравнений
   б. нахождения корня произвольной функции
   в. перемножения многозначных чисел
   г. оценки вычислительной сложности рекурсивных алгоритмов

8. Рекуррентное уравнение для нахождения вычислительной сложности обхода бинарного дерева записывается так

а. $T(n) = T\left(\frac{n}{2}\right) + O(1)$

б. $T(n) = 2T\left(\frac{n}{2}\right) + O(1)$

в. $T(n) = 7T\left(\frac{n}{2}\right) + O(n^2)$

г. $T(n) = 2T\left(\frac{n}{2}\right) + O(\log n)$

9. Рекуррентное уравнение для нахождения вычислительной сложности сортировки слиянием записывается так

а. $T(n) = T\left(\frac{n}{2}\right) + O(1)$

б. $T(n) = 2T\left(\frac{n}{2}\right) + O(1)$

в. $T(n) = 7T\left(\frac{n}{2}\right) + O(n^2)$

г. $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$

10. Функция изоэффективности $W = K(pT_p - W)$ для параллельного алгоритма определяет

а. условие для возникновения сверхлинейного ускорения

б. зависимость размера решаемой задачи от числа используемых процессоров для обеспечения постоянного уровня эффективности параллельных вычислений

в. зависимость эффективности от числа процессоров

г. оценку вычислительной сложности рекурсивных алгоритмов

Правильные ответы, вариант 2

Вар. ответа	Номер вопроса									
	1	2	3	4	5	6	7	8	9	10
а)	Х		Х							
б)		Х						Х		Х
в)				Х	Х	Х				
г)							Х		Х	