Краткая теория для 1 лабы по комбинаторным алгоритмам

1. Очередь – упорядоченный набор данных (структура данных), добавление элементов в которых происходит с конца, а изъятие – с начала очереди. Обычно объявляются два указателя соответственно на начало и на конец очереди.
2. Стек – упорядоченный набор данных (структура данных), доступ к которому осуществляется только с одного конца, называемого вершиной стека. Для организации стека можно обойтись всего лишь одним указателем на его начало (вершину).
3. Дек – упорядоченный набор данных (структура данных), представляющий собой двустороннюю очередь, элементы в которую можно добавлять и извлекать с любого конца.
4. Моделирование очереди с помощью двух стеков – один стек будет служить концом очереди (сюда помещаются новые элементы), другой – началом очереди (отсюда элементы будут извлекаться). Так, при попытке извлечения элемента из второго стека проверяем его на пустоту – если пуст, то сначала копируем содержимое первого стека в обратном порядке, в противном случае сразу извлекаем. При попытке помещения элемента в первый стек дополнительных действий не требуется.
5. Односвязный список – структура данных, представляющая собой упорядоченный набор элементов, каждый из которых содержит указатель на следующий. В отличие от массива, односвязный список – структура последовательного доступа, а не произвольного.
6. Двусвязный список – структура данных, представляющая собой упорядоченный набор элементов, каждый из которых содержит указатель на следующий и предыдущий элемент.
7. f = O(g) если существует такая константа, что верно f<=cg
8. f = Ω(g) если существует такая константа, что верно f>=cg
9. f = Θ(g) если существуют две константы, для которых верно c1g<=f<=c2g
10. f = o(g) если для всех констант верно f<=cg
11. f = ω(g) если для всех констант верно f>=cg
12. Сложность алгоритма в худшем случае – наибольшее количество операций, задаваемых алгоритмом на всех входах определенной размерности.
13. Сложность алгоритма в среднем случае – среднее количество операций, задаваемых алгоритмом на всех входах определенной размерности.
14. Временная сложность алгоритма – асимптотическая оценка функции трудоемкости алгоритма в худшем случае.
15. Пространственная сложность алгоритма – асимптотическая оценка функции объема памяти алгоритма в худшем случае.
16. Трудоемкость алгоритма – количество элементарных операций, которые учитываются при анализе алгоритма.
17. Объем памяти – максимальное количество ячеек памяти, задействованных в ходе работы алгоритма для входа определенной размерности.
18. Вычисление чисел Фибоначчи (экспоненциальный рекурсивный алгоритм):  
    Fib (n)  
     If n==0 return 0  
     If n==1 return 1  
     return Fib(n-1) + Fib(n-2)
19. Вычисление чисел Фибоначчи (полиномиальный алгоритм):  
    Fib(n)  
     fibs[0]=0  
     fibs[1]=1  
     for i=2 to n do  
     fibs[i] = fibs[i-1] + fibs[i-2]  
    return fibs[n]
20. Количество перестановок равно n!, если множество не является мультимножеством и n!/(n1!n2!ni!) в противном случае где n=n1+n2+…+ni – длина перестановки, а ni – количество повторений соответствующего элемента в ней.
21. Количество сочетаний без повторений n!/(k!(n-k)!) и с повторениями (n+k-1)!/(k!(n-1)!)
22. Перестановка – любой упорядоченный набор элементов данного множества, среди которых нет одинаковых.
23. Инверсия – такая пара элементов перестановки xi и xj, что i<j и xi>xj
24. Обратная перестановка получается, если в таблице исходной перестановки поменять местами строки и отсортировать по верхней строке.
25. Таблица перестановок – верхняя строка – ключ, нижняя строка – значения
26. Матричное представление перестановки – квадратная бинарная матрица, в каждой строке которой содержится один единичный элемент номер которого соответствует тому элементу, номер которого указан в столбце таблицы перестановки с номером текущей строки.
27. Рекуррентное соотношение – разбиение задачи на более мелкие подобные подзадачи, решить которые, как правило, намного проще.
28. Алгоритм сортировки пузырьком:
    1. Худший случай: Число сравнений – n(n-1)/2, число обменов – n(n-1)/2
    2. Средний случай: Число сравнений – n(n-1)/2, число обменов – порядка n2
    3. Устойчивость: устойчивый
29. Алгоритм сортировки вставками:
    1. Худший случай: Число сравнений – n(n-1)/2, число обменов – (n+1)(n-1)/2
    2. Средний случай: Число сравнений – порядка n2, число обменов – порядка n2
    3. Устойчивость: устойчивый
30. Алгоритм сортировки бинарными вставками:
    1. Худший случай: Число сравнений – nlog(n), число обменов - (n+1)(n-1)/2
    2. Средний случай: Число сравнений – порядка nlog(n), число обменов – порядка n2
    3. Устойчивость: устойчивый
31. Алгоритм сортировки выбором:
    1. Худший случай: Число сравнений – n(n-1)/2, число обменов – n-1
    2. Средний случай: число сравнений – n(n-1)/2, число обменов – n-1
    3. Устойчивость: устойчивый