**Вариант 1**

**1) Может ли матрица  быть матрицей смежности неориентированного графа?[1]**

**Ответ:** *нет*

**2) Является ли следующее утверждение правильным: если граф неориентированный, то матрица смежности симметричная; [1]**

**Ответ:** *да*

**3)** **Можно ли реализовать операцию INSERT над динамическим множеством в однократно связанном списке так, чтобы время ее работы было равно О(1)? [1] (*Ответ пояснить*)**

**Ответ:** *Да, т.к. мы имеем указатель на начало (конец) односвязного списка. Следовательно, можем просто добавить элемент слева (справа).*

**4)** **Объясните, как с помощью одного массива А[1..n] можно реализовать два стека таким образом, чтобы ни один из них не переполнялся, пока суммарное количество элементов в обоих стеках не достигнет n. Опера­ции Push и Pop должны выполняться в течение времени 0(1). [1]**

**Ответ:** *Стеки должны расти с концов массива навстречу друг другу: первый должен занимать места*

*Содержание[1] ... Содержание[Длина1],*

*а второй -*

*Содержание[n] ... Содержание[n - Длина2 + 1]*

*(вершины обоих стеков записаны последними).*

**5) Очередь позволяет добавлять элементы с од­ного конца, а извлекать — с другого. *Очередь с двусторонним доступом,* или *дек* (deque), предоставляет возможность производить вставку и уда­ление с обоих концов. Напишите процедуру, выполняющуюся в течение времени О(1) и позволяющую вставлять элементы с начала дека, реализованного с помощью массива. [2]**

***Ответ:*** *ErrType l\_insert(deq \*d, elementtype \*et) {
 celltype \*oldct;

 if (d->size == MAX\_DEQ\_SIZE) return bErrDeqIsFull;

 if (d->size != 0) {
 oldct = d->left;
 d->left->left = cellalloc(NULL, d->left);
 if (leftct(d->left) == NULL) return bErrMemory;
 d->left = leftct(d->left);
 d->left->right = oldct;
 }

 memcpy(&(d->left->element), et, sizeof(elementtype));
 (d->size)++;

 return bErrOk;
}*

**6) Очередь позволяет добавлять элементы с од­ного конца, а извлекать — с другого. *Очередь с двусторонним доступом,* или *дек* (deque), предоставляет возможность производить вставку и уда­ление с обоих концов. Напишите процедуру, выполняющуюся в течение времени О(1) и позволяющую удалять элементы с начала дека, реализованного с помощью массива. [2]**

**Ответ:** #define leftct(a) (celltype \*)(a->left)
#define rightct(a) (celltype \*)(a->right)

ErrType l\_delete(deq \*d) {

 if (d->size == 0) {
 return bErrdeqIsNull;
 }

 if (d->size != 1){
 d->left = rightct(d->left);
 cellfree(leftct(d->left));
 d->left->left = NULL;
 }

 (d->size)--;

 return bErrOk;
}

**7) Чему равно наименьшее и наибольшее число ребер в связном гра­фе без петель и кратных ребер с *n* вершинами? [1]**

**Ответ:** *Наименьшее – n-1 (дерево), наибольшее – n(n-1)/2 (полный граф)*

**8) Какая модель теории графов адекватна следующей задаче: Требуется построить схему электрической сети, в которой клеммы должны быть соединены с помощью проводов наименьшей общей длины. [1]**

**Ответ:** *Нахождение минимального остовного дерева.*

**Вариант 2**

**1) Является ли следующее утверждение правильным: если матрица смежности несимметричная, то граф ориентированный; [1]**

**Ответ:** *Да*

**2) Является ли следующее утверждение правильным: если диагональные элементы матрицы смежности – нули, то граф неориентированный? [1]**

**Ответ:** *Нет*

**3) Можно ли реализовать операцию DELETE над динамическим множеством в однократно связанном списке так, чтобы время ее работы было равно О(1)? [1] (*Ответ пояснить*)**

**Ответ:** *Если мы знаем позицию элемента, то можно. Но если нам нужно удалить элемент с заданным ключом, то в наихудшем случае понадобится время O(n) поскольку сначала будет необходимо найти элемент по этому ключу.*

**4) Продемонстрируйте вставку ключей 5, 28, 19, 15, 20, 33, 12, 17, 10 в хеш- таблицу с разрешением коллизий методом цепочек. Таблица имеет 9 ячеек, а хеш-функция имеет вид h(k) = к mod 9. [1]**

**Ответ:**

|  |  |
| --- | --- |
| 0 |  |
| 1 | 29,19,10 |
| 2 | 20 |
| 3 | 2 |
| 4 | 5 |
| 5 |  |
| 6 | 15,33 |
| 7 |  |
| 8 | 17 |

**5) Очередь позволяет добавлять элементы с од­ного конца, а извлекать — с другого. *Очередь с двусторонним доступом,* или *дек* (deque), предоставляет возможность производить вставку и уда­ление с обоих концов. Напишите процедуру, выполняющуюся в течение времени О(1) и позволяющую вставлять элементы с конца дека, реализованного с помощью массива. [2]**

**Ответ:** ErrType r\_insert(deq \*d, elementtype \*et) {
 celltype \*oldct;

 if (d->size == MAX\_DEQ\_SIZE) return bErrDeqIsFull;

 if (d->size != 0) {
 oldct = d->right;
 d->right->right = cellalloc(d->right, NULL);
 if (rightct(d->right) == NULL) return bErrMemory;
 d->right = rightct(d->right);
 d->right->left = oldct;
 }

 memcpy(&(d->right->element), et, sizeof(elementtype));
 (d->size)++;

 return bErrOk;
}

**6) Очередь позволяет добавлять элементы с одного конца, а извлекать — с другого. *Очередь с двусторонним доступом,* или *дек* (deque), предоставляет возможность производить вставку и уда­ление с обоих концов. Напишите процедуру, выполняющуюся в течение времени О(1) и позволяющую удалять элементы с конца дека, реализованного с помощью массива. [2]**

**Ответ:** #define leftct(a) (celltype \*)(a->left)
#define rightct(a) (celltype \*)(a->right)

ErrType r\_delete(deq \*d) {

 if (d->size == 0) {
 return bErrdeqIsNull;
 }

 if (d->size != 1){
 d->right = leftct(d->right);
 cellfree(rightct(d->right));
 d->right->right = NULL;
 }

 (d->size)--;

 return bErrOk;
}

**7) Чему равно наименьшее и наибольшее число ребер в графе без петель и кратных ребер с *n* вершинами? [1]**

**Ответ:** *Наименьшее – 0 (несвязный граф), наибольшее – n(n-1)/2 (полный граф)*

**8) Какая модель теории графов адекватна следующей задаче:**

**Имеется сеть связи, соединяющая *n* узлов. Если выйдут из строя некоторые каналы, то связь между узлами может быть нарушена. Какие каналы можно удалить без нарушения связи? Какие каналы нужно удалить, что­бы связь не нарушалась, а общая стоимость всех каналов была минимальной? [1]**

**Ответ:** *Нахождение минимального остовного дерева.*