Diccionarios.

ED Grupo A. Profesor: Isabel Pita.

Nombre del alumno:

1. (1 punto) Indica el coste de las siguientes operaciones sobre los tipos `map` y `unordered_map` de la librería `STL` en relación al número n de elementos del diccionario.

operación	`map`	`unordered_map`
`count(clave)`		
`insert({c,v})`		
`erase(clave)`		

Respuesta:

operación	`map`	`unordered_map`
`count(clave)`	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$
`insert({c,v})`	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$
`erase(clave)`	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$

2. (1 punto) Define el *factor de carga* de una tabla hash. Indica para que se utiliza y el intervalo de valores que suele tomar.

Respuesta:

   a) El factor de carga es el numero de posiciones ocupadas respecto al tamaño de la tabla. Se define como:
   $fc = \frac{\texttt{posiciones ocupadas}}{\texttt{tamaño de la tabla}}$
   b) Indica cuando se debe ampliar la tabla por estar saturada.
   c) Normalmente toma valores entre 0,7 y 0,8.

3. (0,5 puntos) ¿Cuántos valores puede guardar cómo máximo una tabla hash con factor de carga 0,7 si el tamaño de la tabla es 1009?.

Respuesta:

$1009 * 0,7 = 706,3$, por lo tanto se pueden guardar como máximo 706 valores, si se quieren guardar más valores la tabla se redimensiona.

4. (1 punto) Dibuja una tabla hash abierta, con clave de tipo `int` y valor de tipo `string`, tamaño de tabla 11, y función hash el valor de la clave. La tabla tendrá las claves: 1,9,12,22,44,38,21. Los valores asociados no son relevantes.

Respuesta:

5. (1 punto) Indica cómo se comportan las siguientes operaciones en caso de que la clave exista en el diccionario y en caso de que no exista.

operación	La clave está en la tabla	La clave no está en la tabla
`t.insert({c,v})`		
`t[clave]`		
`t.at(clave)`		

Respuesta:

operación	La clave está en la tabla	La clave no está en la tabla
`t.insert({c,v})`	Devuelve `false` y un iterador al elemento de la tabla. La tabla no se modifica.	Devuelve `true` y un iterador al nuevo elemento. Se añade el elemento a la tabla.
`t[clave]`	Devuelve una referencia al elemento de la tabla.	Añade el elemento a la tabla y devuelve una referencia al elemento.
`t.at(clave)`	Devuelve una referencia al elemento de la tabla.	Lanza una excepción.

6. (1 punto) Dibuja una tabla hash cerrada, con clave de tipo `int` y valor de tipo `string`, tamaño de tabla 11, función hash el valor de la clave, y método de resolución de colisiones la siguiente posición libre. En la tabla se han realizado las siguientes operaciones:

```
t.insert({22,"a"});
t.insert({44,"f"});
t.insert({1,"d"});
t.erase(44);
t.insert({12,"v"});
t.erase(22);
t.insert({15,"g"});
t.insert({6,"l"});
```

```
t.insert({4,"g"});
t.erase(15);
t.insert({16,"u"});
t.insert({5,"c"});
t.insert({7,"g"});
```

Respuesta:

0	22	a	liberado
1	12	j	ocupado
2	1	d	ocupado
3			Libre
4	15	g	liberado
5	4	g	ocupado
6	6	l	ocupado
7	16	u	ocupado
8	5	c	ocupado
9	7	g	ocupado

7. (1 punto) La tabla anterior ha sobrepasado su factor de carga. Indica el resultado de ampliar la tabla. Justifica tu respuesta.

Respuesta:

El tamaño de la nueva tabla es el primer primo posterior al doble del tamaño actual. El doble del tamaño actual es 22 y el primer primo posterior es el 23. Por lo tanto el tamaño de la nueva tabla es 23.

0			Libre
1	1	d	ocupado
2			Libre
3			Libre
4	4	g	ocupado
5	5	c	ocupado
6	6	l	ocupado
7	7	g	ocupado
8			Libre
9			Libre
10			Libre
11			Libre
12	12	v	ocupado
13			Libre
14			Libre
15			Libre
16	16	u	ocupado
17			Libre
18			Libre
19			Libre
20			Libre
21			Libre
22			Libre

8. (0,5 puntos) Que problema aparece en las tablas hash cerradas que tienen como método de resolución de colisiones el buscar la primera posición libre?

Respuesta:

Los valores tienden a agruparse formando bloques de claves consecutivas. Esto degrada la tabla, ya que las operaciones dentro de cada bloque tienen coste lineal en el tamaño del bloque.