Examen Final del Taller - Regulares

Algoritmos y Estructuras de Datos II

Tarea

El alumno deberá implementar el tipo abstracto de dato *Skyline* (ver especificación abajo) en el lenguaje de programación C, utilizando la técnica de ocultamiento de información visto en el taller de la materia.

Es requisito mínimo para aprobar lo siguiente:

- Implementar en C el TAD *skyline* (utilizando punteros a estructuras y manejo dinámico de memoria).
- Implementar en C una interfaz de línea de comando para que usuarios finales puedan usarlo (ver detalles más abajo).
- El programa resultante no debe tener *memory leaks* ni accesos inválidos a memoria, se chequeará tal condición usando valgrind.

Especificaciones

Se define el TAD *skyline*, que representa la silueta que dejan los edificios de una ciudad en el horizonte.

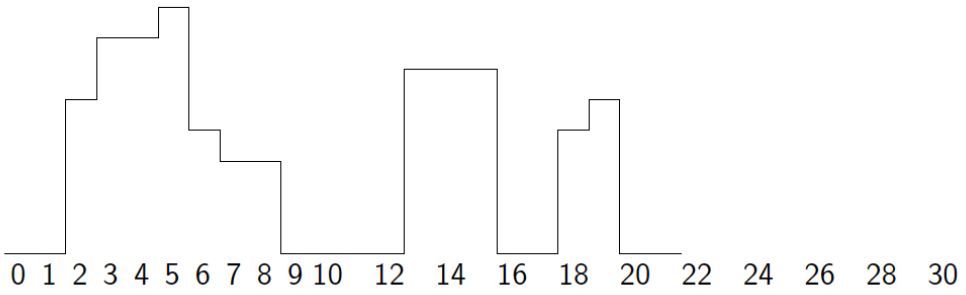


Figura 1: Ejemplo de *skyline*

El tipo abstracto tiene 2 constructores:

- uno para crear la silueta vacía, es decir, sin edificios
- otro para agregar un edificio a la izquierda de una silueta preexistente.

Este último constructor tiene por argumento dos naturales a y h, y una silueta s, y agrega a la izquierda de s un edificio de ancho a y altura h

TAD skyline

Constructors a implementar

empty : skyline

add_building : natural × natural × skyline → skyline

operaciones a implementar

overlap : skyline × skyline → skyline

building : natural × natural × natural → skyline

overlap_building : natural × natural × natural × skyline → skyline

skyline_max_height : skyline → natural

area : skyline → natural

ecuaciones

add_building(a, h, add_building(b, h, s)) = add_building(a+b, h, s)

add_building(0, h, s) = s

overlap(empty, s) = s

overlap(s, empty) = s

a > b > 0 => overlap(add_building(a, h, s), add_building(b, k, t)) = add_building(b, max(h, k), overlap(add_building(a-b, h, s), t))

b > a > 0 => overlap(add_building(a, h, s), add_building(b, k, t)) = add_building(a, max(h, k), overlap(s, add_building(b-a, k, t)))

a = b > 0 => overlap(add_building(a, h, s), add_building(b, k, t)) = add_building(a, max(h, k), sup(s, t))

building(d, a, h) = add_building(d, 0, add_building(a, h, empty))

overlap_building(d, a, h, s) = overlap(building(d, a, h), s)

Así, la silueta de la Figura 1 puede ser construida como:

```
s = add_building(2, 0, add_building(1, 5, add_building(2, 7,
                        add_building(1, 8, add_building(1, 4, add_building(2, 3,
                        add_building(4, 0, s'))))))))
```

donde s' es:

```
s' = add_building(3, 6, add_building(2, 0, add_building(1, 4, add_building(1, 5,
                          add_building(2, 0, empty)))))
```

(la utilización de s' se debe únicamente a intentar facilitar la lectura de la definición de s).

Una silueta puede construirse de diferentes maneras, por ejemplo lo siguiente es otra de construir s':

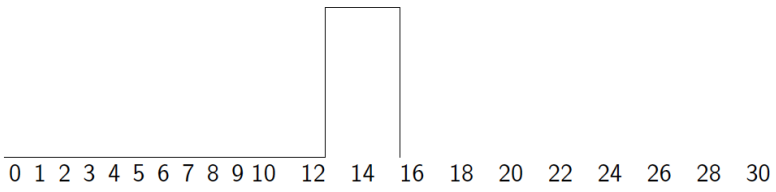
```
s' = add_building(1, 6, add_building(2, 6, add_building(2, 0, add_building(1, 4,
                          add_building(1, 5, add_building(2, 0, empty))))))
```

La primera ecuación del TAD dice justamente que dos edificios adyacentes de igual altura impactan en la silueta igual que un edificio suficientemente ancho.

La segunda ecuación dice que un edificio de ancho 0 no afecta la silueta. No así un edificio de altura 0 pero ancho positivo, ya que puede funcionar como separador de dos edificios.

La operación building se utiliza para construir la silueta de un edificio solo.

Por ejemplo, building(13, 3, 6) es la silueta:



La operación overlap permite superponer dos siluetas. La silueta del primer ejemplo se puede obtener superponiendo las siguientes dos siluetas:

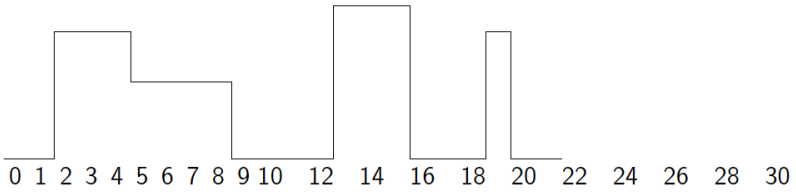
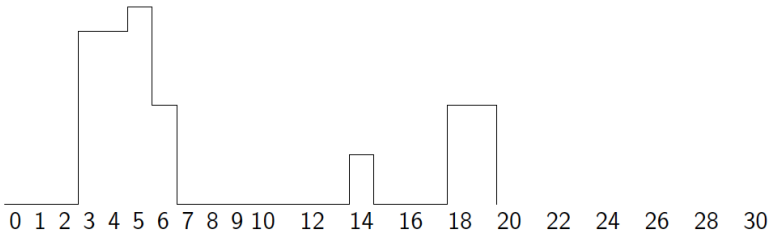


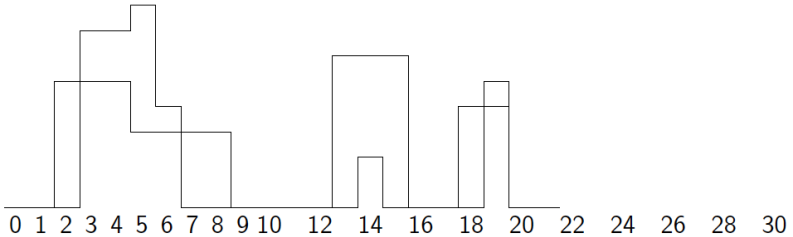
Figura 2:



Figura

3: En efecto, al superponerlas "visualmente" se

obtiene:



que, ignorando las líneas internas que solo se dejaron para que se aprecie la superposición de

las siluetas de las Figuras 2 y 3, nos da la silueta de la Figura 1.

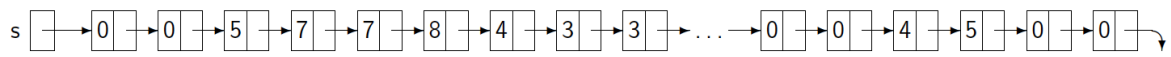
Finalmente, la operación max height devuelve la máxima altura de un edificio en la silueta (en la Figura 1, este valor es 8); la operación area devuelve la superficie por debajo de la silueta (para la Figura 1 sería 64).

El objetivo del examen es implementar el TAD *skyline* con una lista enlazada según la siguiente definición. No es necesario implementar la lista enlazada como TAD separado.

Se utiliza el nombre skyline para la representaci´on de la silueta, como se muestra a continuaci´on:

```
type node = tuple
              height: nat
              next: pointer to node
            end
  type skyline = pointer to
            node
```

Con esta representaci´on, la silueta s de la Figura 1 se podr´ıa implementar as´ı:



Algunos puntos a tener en cuenta:

- Recordar que la estructura especificada es una estructura teórica, y que al momento de escribir el código y cumplir con los requerimientos solicitados puede ser necesario extender la respectiva estructura en C.
- Al implementar la lista enlazada pensar donde conviene agregar los nuevos elementos (al principio? al final?).
- La implementación de la operación overlap no modifica las siluetas que recibe como argumento sino que debe generar una silueta nueva como resultado.
- Las operaciones max height y area deben tener **orden constante** en la implementación.

Importante: la especificaci´on te´orica del TAD no incluye un destructor del tipo, pero la implementaci´on en C si debe proveerlo (y la interfaz de línea de comando debería usarlo).

La interfaz de lınea de comandos

Se deberá proveer además una interfaz de línea de comandos para manipular siluetas. Al inicio del programa deberán existir dos siluetas vacıas, una primaria y otra secundaria (pensar por que y para que, en función de las operaciones listadas abajo).

Las opciones a presentar al usuario final son:

- Agregar un edificio a la silueta primaria. Solicita al usuario dos valores enteros positivos a y h que representan el ancho y la altura del edificio a agregar.
- Agregar un edificio a la silueta secundaria. Agrega un edificio como en el punto anterior, pero a la silueta secundaria.
- Superponer las siluetas actuales. Superpone las siluetas primaria y secundaria e imprime por pantalla el resultado (ver punto siguiente sobre dump/impresión).

- Imprimir la silueta primaria por pantalla. Muestra por la salida estándar una representación simple de la silueta primaria que se está manipulando (no implica necesariamente imprimir el dibujo de la silueta en sı como en las figuras de este enunciado, sino que con una representación visual de la lista interna alcanza). Imprime además la altura máxima y el área.

- Imprimir la silueta secundaria por pantalla. Idéntico al punto anterior, pero para la silueta secundaria.
- Salir de la aplicación.

Para interactuar con el usuario final, se puede usar la funci´on auxiliar `readline from stdin` del módulo `helpers`, que será entregado junto con este enunciado.