Práctica 6 : Secuencias

1. Los números de Fibbonacci son una secuencia de enteros dados por la siguiente recurrencia[1]:

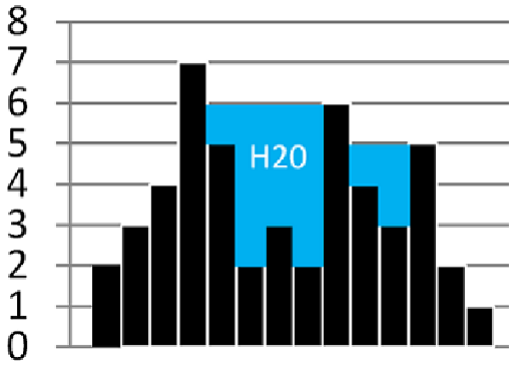
$$\begin{aligned} F_{-1} &= 1 \\ F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \end{aligned}$$

Usando esta definición es posible probar por inducción la siguiente propiedad:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}$$

Dar una definición de la función $fibSeq : Nat \to Seq\ Nat$, que dado un natural n calcule la secuencia de los primeros n números de Fibonnacci, cuyo trabajo y profundidad sean $W(n) = n$ y $S(n) = lg(n)$. Utilizar la función *scan*.

2. Considerar el siguiente problema: Si se vierte agua sobre un histograma, ¿cuánta agua (en términos del área) queda almacenada sobre el mismo? El siguiente dibujo puede servir como ilustración del problema dado.



El problema puede resolverse calculando la cantidad de agua que queda almacenada sobre cada barra del histograma. Por ejemplo, para una barra b_i la cantidad de agua que se acumula sobre ésta es igual al máximo entre 0 y

$$min(maxL, maxR) - altura(b_i)$$

donde $maxL$ es el valor máximo entre las alturas de las barras que están a la izquierda de b_i y $maxR$ es el valor máximo entre las alturas de las barras que están a la derecha de b_i. Para el histograma de la figura el resultado sería 15.
Definir una función *aguaHist* que dada una secuencia de enteros (que represente un histograma) devuelva la cantidad de agua almacenada, utilizando las funciones *scan* y *reduce*.

3. El problema de determinar si en una expresión los paréntesis están bien anidados (todo paréntesis que se abre se cierra, y en ningún prefijo hay más cerrados que abiertos), puede resolverse utilizando una función $matchParen : Seq\ Paren \to Bool$, que dada una secuencia del tipo:

data *Paren* = *Open* | *Close*

(que representa los caracteres `'('` y `')'`), devuelva *True* si la secuencia contiene la misma cantidad de valores *Open* que *Close* y además cada prefijo de la secuencia no contiene menos valores *Open* que *Close*, y *False* en caso contrario.

[1] La definición del elemento (-1)-ésimo de la secuencia de Fibbonacci fue necesaria para que la propiedad dada sea cierta. La elección de este número no afecta los demás números de la secuencia.

a) Dar una definición de *matchParen* que implemente un algoritmo *Divide & Conquer*. La función *showT* del TAD de secuencias puede ser de utilidad para dividir una secuencia en dos.

Ayuda: Definir una función $matchParen' : Seq\ Paren \to (Int, Int)$ tal que si $matchParen'\ s = (i, j)$ entonces s puede reducirse a $)^i(^j$ eliminando todas las subsecuencias de la forma (). La función *matchParen* quedaría definida en términos de esta función como:

$$matchParen\ s = matchParen'\ s \equiv (0, 0)$$

b) Dar una definición de *matchParen* que utilice la función *scan*. Comparar los costos (trabajo y profundidad) de las dos versiones.

4. Se desea encontrar la distancia mínima entre dos puntos de un plano de n puntos. Un algoritmo básico, pero ineficiente, consiste en calcular las distancias entre los $\binom{n}{2}$ pares de puntos y elegir la mínima. Aquí no se aprovecha la estructura que posee un plano de 2 dimensiones.
Un algoritmo *Divide & Conquer* más eficiente consiste en, dada una secuencia S de puntos del plano, ordenada según la primer componente:

- Dividir S en dos secuencias de puntos: S_1 y S_2 con cantidad de puntos aproximadamente iguales.
- Calcular la distancia mínima entre dos puntos de S_1 (d_1) y dos puntos de S_2 (d_2).
- Obtener la distancia mínima de dos puntos de S utilizando los valores d_1 y d_2.

Definir una función $closestPair : Seq\ (Int, Int) \to Nat$ que calcule la distancia Euclíidea mínima entre dos puntos de una secuencia de puntos dada, cuyo trabajo esté definido por la siguiente recurrencia:

$$W_{closestPair}(n) = 2W_{closestPair}(\frac{n}{2}) + kn + W_{combine}(n)$$

donde $W_{combine}(n)$ es el trabajo realizado por la función que lleva a cabo el último paso del algoritmo.
Recordar que la distancia Euclídea entre dos puntos $p = (x, y)$ y $q = (w, z)$ es $\sqrt{(x-w)^2 + (y-z)^2}$.

5. Dada una secuencia de enteros, el problema conocido como "la subsecuencia contigua creciente más larga" consiste en encontrar la cantidad mayor de crecimientos contiguos en una secuencia.
Por ejemplo,

$$sccml\ \langle 9, 3, 5, 1, 2, 3, 4, 5, 6, 1\rangle = 5$$
$$sccml\ \langle 5, 6, 2, 3, 5, 1, 9\rangle = 2$$
$$sccml\ \langle 1, 4, 6, 7, 8, 9, 10, 3\rangle = 4$$

Definir una función $sccml : Seq\ Int \to Int$, que resuelva el problema utilizando la función *scan*.

6. Definir las siguientes operaciones que permiten extender el TAD de secuencias.

a) $merge : (a \to a \to Ordering) \to Seq\ a \to Seq\ a \to Seq\ a$, que dadas una relación de orden y dos secuencias ordenadas (respecto a ésta relación) s_1 y s_2, construye una secuencia ordenada con los elementos de s_1 y s_2.

b) $sort : (a \to a \to Ordering) \to Seq\ a \to Seq\ a$, que ordena una secuencia seguń una relación de orden dada.

c) $maxE : (a \to a \to Ordering) \to Seq\ a \to a$, que devuelve el máximo de una secuencia.

d) $maxS : (a \to a \to Ordering) \to Seq\ a \to Nat$, que devuelve el índice de un máximo en la secuencia.

e) $group$:$(a \to a \to Ordering) \to Seq\ a \to Seq\ a$, que dada una secuencia agrupa los elementos iguales contiguos. Por ejemplo,

$$group\ \langle 1, 1, 2, 3, 4, 4, 2, 2\rangle = \langle 1, 2, 3, 4, 2\rangle$$

f) $collect : Seq\ (a, b) \to Seq\ (a, Seq\ b)$, que recolecta todos los datos asociados a cada clave y devuelve una secuencia de pares ordenada según el primer elemento. Por ejemplo,

$$collect\ \langle(2, \texttt{"A"}), (1, \texttt{"B"}), (1, \texttt{"C"}), (2, \texttt{"D"})\rangle = \langle(1, \langle\texttt{"B"}, \texttt{"C"}\rangle), (2, \langle\texttt{"A"}, \texttt{"D"}\rangle)\rangle$$

7. Una universidad cuenta con una base de datos de los estudiantes que rindieron los exámenes de ingreso y las notas de las evaluaciones realizadas. Un estudiante ingresará la universidad si el promedio de los resultados de los exámenes es mayor o igual a 70 y quedará en lista de espera si el promedio es mayor a 50 y menor a 70. Se desea saber cuántos estudiantes aprobaron el ingreso, cuántos quedaron en lista de espera y cuántos no podrán ingresar a la universidad, junto con las notas máxima de cada uno de los 3 casos.
Definir una función $datosIngreso : Seq\ (String, Seq\ Int) \to Seq\ (Int, Int)$ que dada una secuencia con los nombres de los estudiantes y las notas de los exámenes, calcule los datos necesarios (cantidad de alumnos, nota máxima) de los exámenes de ingreso. Definirla en términos de *mapCollectReduce*.

8.

a) Definir una función $countCaract : Seq\ (Seq\ Char) \to Seq\ (Char, Int)$ que dada una colección de textos calcule la cantidad de veces con que aparecen los caracteres en los textos. Definirla en términos de *mapCollectReduce*.

b) Definir una función $huffman : Seq\ (Seq\ Char) \to Seq\ (Int, Seq\ Char)$ que dada una colección de textos calcule las frecuencias con que cada caracter aparece en los textos. La secuencia resultado debe contener pares de la forma $(n, caracteres\ con\ frecuencia\ n)$ y debe estar ordenada según las frecuencias de los caracteres. Utilizar las funciones *countCaract*, *map* y *collect*.