Tarea 6.

Leonardo Brambila Ayala.

Lenguajes de programación
Lic. en Ciencias de la Computación
Universidad de Sonora
Semestre 2023-1

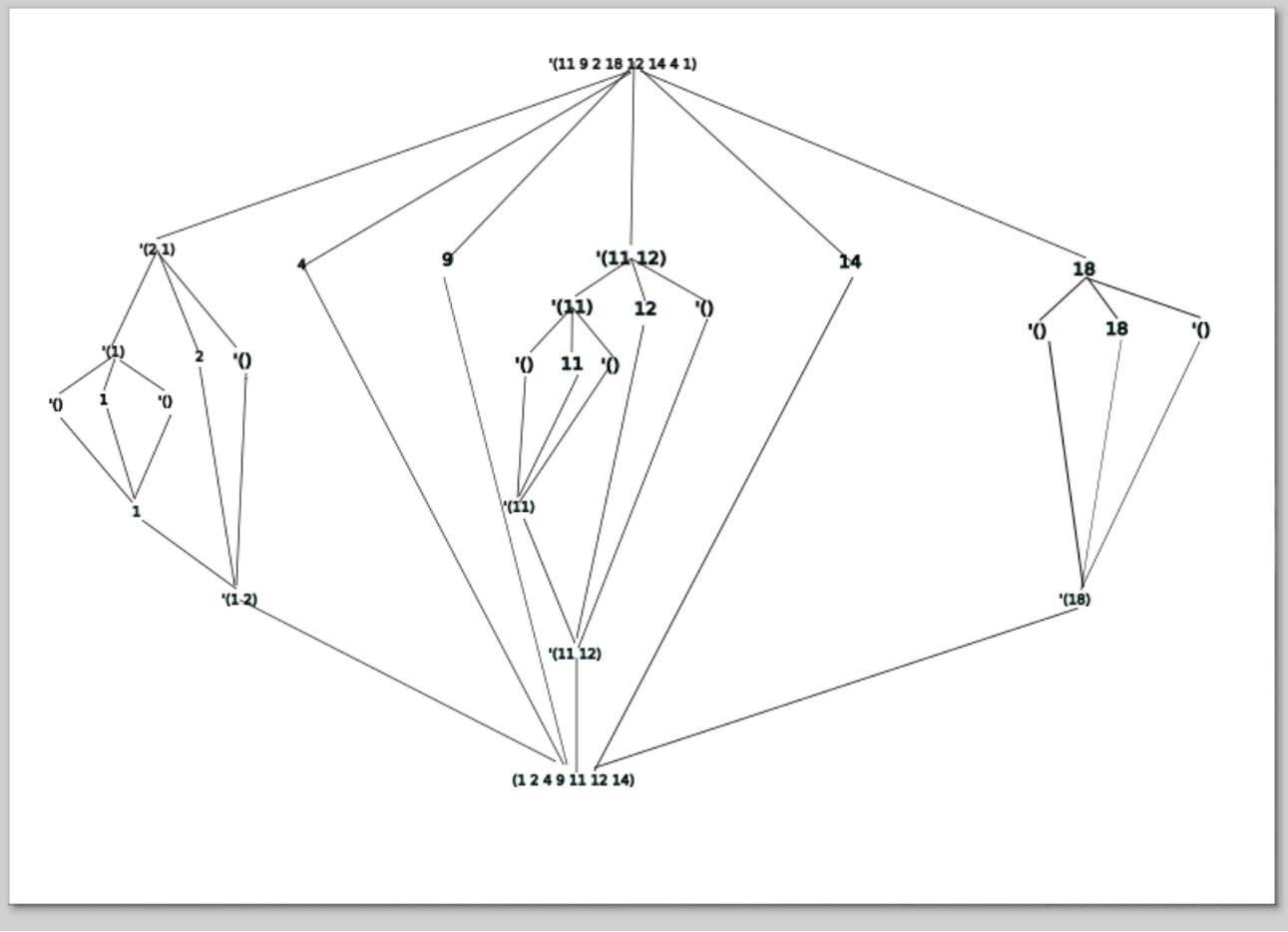
1 Recursividad generativa.

1.1 Recursión sin estructura.

Problema 4: La llamada (bundle '("a" "b" "c") 0) es un buen uso de bundle? ¿qué produce? ¿por qué?
R: Si es un buen uso; produce una lista vacia **'()**; Porque, en el codigo se verifica si "s" esta vacia o "n"(representa la longitud de los subconjuntos de la lista original) es igual a cero; ya que sabemos que "s" no esta vacia, pero "n" si es cero, entonces lo que produce seria una lista vacia **'()**, esto pasa, porque no agrupara ningun elemento

1.2 Recursión que ignora estructura.

Problema 10: Si la entrada a quicksort contiene varias repeticiones de un número, va a regresar una lista estrictamente más corta que la entrada. Responde el por qué y arregla el problema.



1.3 Adaptación de los principios de diseño.

Problema 15: Repasa tus soluciones a los problemas anteriores y comenta en cada función su firma, qué es lo que hace y cómo lo hace. En caso de ser recursión estructural, el cómo puede ser breve ya que la recursión debe naturalmente modelarse sobre la estructura de la entrada. Puedes inspirar tus firmas en los contratos de Racket.

1.4 Terminación.

Problema 17: Considera la siguiente definición de smallers, uno de los procedimientos utilizados en quicksort, responde en qué puede fallar al utilizar la siguiente versión modificada en el procedimiento de ordenamiento.
R= El problema de esta implementacion es que no elimina elementos que sean mayores a 'n', ya que en el quicksort, se divide en tres partes, dividir el mayores, los menores y los iguuales ; pero en esta implementacion solo cubre los elementos menores e iguales al pivote.

Problema 18: Problema 18: ¿En qué casos el subproblema no es estructuralmente más pequeño que el problema original en la definición de la función de Ackermann?

$$A(m,n) = \begin{cases} n+1, & \text{si } m = 0; \\ A(m-1,1), & \text{si } m > 0 \text{ y } n = 0; \\ A(m-1, A(m,n-1)), & \text{si } m > 0 \text{ y } n > 0 \end{cases}$$

R= Cuando 'm' es igual a cero, ya que no es posible una reduccion de 'm', ya que esta no se llama recursivamente en el caso de "m - 1", entonces en conclusion, este esun caso en el que el subproblema no es mas pequeño que 'm'.

1.5 Tomando decisiones.

Probema 19: Describe con tus propias palabras cómo funciona find-largest-divisor de gcd-structural. Responde por qué comienza desde (min n m).

R = Primeros tenemos la funcion "(define (gcd-structural n m) ...)", recibe dos argumentos 'n' y 'm', en donde dentro de esta funcion se manda a llamar a "(define (find-largest-divisor k)...)", en donde 'k', es igual al valor minimo de entre 'n' y 'm', ya que estamos buscando los divisores mas grandes que 'k', que seria el menor de los dos valores. La funcion "(define (find-largest-divisor k)...)", verifica si el contador "i", es un divisor comun de 'n' y 'm'; y si 'n' y 'm', son divisibles por 'i', entonces este es un divisor comun y se devolvera 'i'.
Si no es el caso pasado, se decrementara 'k' en 1 y se hace una llamada recursiva, con el nuevo valor de 'k', $k = k - 1$, esto se hara hasta que 'k', tenga valor de uno y se devolvera 1, que en este caso seria el divisor comun.

Problema 20: Describe con tus propias palabras cómo funciona find-largest-divisor de gcd-generative.

R = Toma dos argumentos 'max' y 'min', estos son los valores maximos y minimos de 'n' y 'm', esto ayuda a que no estemos buscando los divisores mas grandes que el minimo de estos dos valores.

Si 'min' es igual a 0, si es este caso devuelve 'max' como el divisor mas grande comun, de lo contario, este calcula el residuo de la division de max/min, despues se llama recursivamente al procedimiento (remainder (max,min)), con los nuevos valores de 'max' y 'min'; esta recursion continua diviediendolos y actualizando a 'min', hasta que 'min', sea igual a 0, mientras que 'max', permanece estativo, sin ninguncambio.

Solamente se acaba cuando 'min' sea igual a 0, y si es este caso devolvera max como el divisor mas grande comun.