Mini-Proyecto II: Estructuras de Datos Compactas

Descripción

En este proyecto se explorará en mayor profundidad (y desde el punto de vista de la implementación) la aplicación de algunas de las estructuras de datos compactas más comunes. Para esto se hará uso de la librería SDSL (el código de la librería, así como ejemplos y la documentación acerca de la misma, se puede encontrar en https://github.com/simongog/sdsl-lite).

La tarea se compone de varios ejercicios que nos irán acercando a la solución a un problema complejo denominado series de tiempo de rasters. En dicho problema, se dispone de matrices bidimensionales que representan una propiedad de una región del espacio en un instante de tiempo. En concreto, los datasets proporcionados representan temperatura. Por tanto, el valor de una celda en una matriz representa la temperatura en esa zona del espacio. Cada dataset se compone de 100 matrices, cada una correspondiendo a un instante de tiempo (específicamente, se dispone de una matriz cada hora). Un ejemplo del tipo de información se puede visualizar en http://www.inf.udec.cl/~dseco/guam.mp4

El problema tiene dos propiedades interesantes: localidad espacial (celdas próximas suelen tener valores similares) y localidad temporal (la misma celda en matrices consecutivas, suele tener valores similares). Estas propiedades lo hacen adecuado para proponer estructuras de datos compactas.

Ejercicios

1. Calcular H_0 para cada uno de los datasets.
2. Para cada dataset, leer las matrices por filas (*row-major order*) y almacenar los valores en int_vector<>.
3. Codificar los vectores anteriores de manera que valores consecutivos repetidos (*runs*) se representen una única vez. Para ello se empleará un bitmap b que marcará con 1 el inicio del run y con 0s los elementos repetidos. La secuencia de valores sin repetición se almacenará en un int_vector<>. Ejemplo, v={3,3,3,2,1,5,5,5,7,7,7} se representará como b={1,0,0,1,1,1,0,0,1,0,0} y v'={3,2,1,5,7}.[1]
4. Dada la representación anterior, soportar acceso a la posición i. Para ello, el bitmap se representará como un rrr_vector o un sd_vector (el que mejor espacio obtenga) con soporte para rank. Tener en cuenta que v[i]=v'[rank(b,i)].
5. Sea M_i la matriz en el instante i. Para toda matriz M_i, i>=1 (es decir, todas menos la primera de cada dataset), crear una matriz binaria BM_i de tal forma que $BM_i[x][y]=(M_i[x][y]<>M_{i-1}[x][y])$. Es decir, para cada celda, almacena un 1 si el valor es diferente al valor de la misma celda en el instante anterior, y un 0 en caso contrario. Representar cada una de las matrices resultantes con un k2-tree (sdsl/k2_tree.hpp) y reportar el espacio que suman en total. Nota: el k2-tree no aparece en la *cheat sheet* por haber sido incorporado más tardíamente.
6. [Para equipos de 2 o más personas] Por cada matriz, representar las diferencias con la matriz anterior de manera compacta. Pista: ejercicios 2 y 3. Nota: observar que las diferencias pueden ser negativas y debe proponerse alguna solución a dicho problema.

1 No se espera que esta parte de la representación obtenga buenos resultados. El motivo es que row-major order no es una solución adecuada para preservar la localidad espacial. Existen otros órdenes como Morton-order, más adecuados para esta labor. Sin embargo, se propone *row-major* para simplificar la tarea.

7. [Para equipos de 3 personas] Empleando las estructuras compactas de los ejercicios 5 y 6, implementar sobre ellas una operación que permita obtener el valor $M_i[x][y]$.

Objetivos específicos

Por cada ejercicio se debe:

- Describir brevemente las funcionalidades de la SDSL empleadas.
- Implementar la solución.
- Evaluar experimentalmente la solución reportando el espacio que ocupa y el tiempo (en los ejercicios que así lo ameriten). Para el espacio se recomienda utilizar la visualización que proporciona la propia librería (ver Measuring and Visualizing Space en https://simongog.github.io/assets/data/sdsl-cheatsheet.pdf).

Normas de entrega

- El proyecto se puede realizar individualmente o en equipos de máximo 3 personas. El informe debe reflejar claramente los autores del proyecto.
- La implementación se debe realizar en C++ (para facilitar el uso de la SDSL).

Bonus (opcional)

Este ejercicio es de carácter voluntario y se puede realizar durante lo que resta del curso. Se puede realizar en conjunto con la actividad de lectura de artículo científico. El puntaje obtenido se sumará a la nota final de la asignatura.

Existe una necesidad de medidas que expresen la compresibilidad de un ráster y de una serie de tiempo de rásters. El valor H_0 calculado en el ejercicio 1 no tiene en cuenta localidad espacial ni temporal. Para otros dominios, como texto, existen una gran cantidad de medidas, pero su adaptación a este dominio no es trivial. Por ejemplo, ¿cómo se puede adaptar H_k? Por tanto, la definición de dichas medidas se puede considerar un problema de investigación abierto. Un par de lecturas relacionadas:

- Danny Hucke, Markus Lohrey, Louisa Seelbach Benkner: A Comparison of Empirical Tree Entropies. 232-246 (SPIRE 2020)
- Arezoo Abdollahi, Neil D. B. Bruce, Shahin Kamali, Rezaul Karim: Lossless Image Compression Using List Update Algorithms. SPIRE 2019: 16-34

Se otorgarán puntos extra por las siguientes actividades:

- Proporcionar nuevas referencias bibliográficas (con resumen) que ayuden a definir el marco teórico del problema.
- Proporcionar nuevas ideas que aporten en la definición de medidas de compresibilidad.
- El trabajo es potencialmente publicable, por lo que si alguien desarrolla la idea con suficiente profundidad puede optar directamente al 7 en la asignatura.