NOMBRE: Jose Maria de la Torre

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 —Estructuras de Datos y Algoritmos

Informe 1

1 Funcionamiento del Programa

Mi programa funciona de la siguiente forma:

1. Crear un stack de cores (utilizando el codigo mostrado en ayudantia)
2. Iterar sobre ese stack mientras no este vació. Luego en cada iteración
3. Tomar un Core y avanzar, en búsqueda de otro core. Originalmente mi idea era que la forma en que se mueva sea aleatoria, de esta manera no avanza siempre en la misma direccion. Pero, no pude implementarlo, por lo tanto avanza desde 0 a la cantidad de buildings que se puede mover. Para ver si puede avanzar revisa lo siguiente.
   (a) Que el building al que me quiero mover no sea el mismo en el cual estoy parado
   (b) Si el building esta tomado
   (c) Si el building esta listo
   (d) Que no tenga color y que no sea Core. Luego:
   (e) Si tiene color y es igual al inicial. Los conecto y termina la recesión
   (f) En el caso que no sea core pero puedo moverme a ese building lo conecto y parto de nuevo la recesión para seguir buscando un core.
4. Ahora, si hubo conexión entre core y core, remuevo los dos core del stack y los muevo a un stack de cores conectados.
5. En el caso que no se pudo conectar, saco otro core ya conectado y lo desconecto
6. Ahora que el otro core se desconecto, corro el algoritmo de conectar core con core de nuevo, pero con mas espacio en el mapa
7. Continuo hasta que todos los cores estén conectados

Este fue mi approach al problema. Claramente no pude terminarlo ya que tuve varios problemas que me impedían seguir mejorando mi algoritmo. Claramente este algoritmo tiene varias fallas para resolver el problema. Alguno de los problemas que tiene son

1. Cuando hace backtracking, se devuelve de manera entera, no solo un paso atras
2. No revisa que existan todas las conexiones necesarias (pueden haber espacios en blanco)
3. No esta optimizado para tardar menos tiempo
4. Al ser random, puede que se demore infinto tiempo en resolver
5. Y mas cosas que aun no descubro

2 Tiempos de ejecucion

Mi programa solamente resuelve un mapa (el de 1x1) y lo resuelve en 0.000373 segundos. Para calcular el tiempo utilice http://stackoverflow.com/questions/5248915/excusión-time-of-c-program. Ahora, abordando el problema en si, creo que debería tardarse mucho más. Al ser un problema de Backtracking, lo más probable es que el programa debiese tardar en rangos de $\mathcal{O}(x^2)$. Ya que hay que probar muchísimas combinaciones. Ahora claramente utilizando podas podemos disminuir el tiempo de ejecución.

Obviamente el tiempo deseado es de menos de 10 segundos, que se podria lograr utilizando podas y heuristicas.

Lo principal que podria hacerle a mi programa para que funcione de mejor manera es agregar un stack o implementar de alguna manera un recuerdo de los caminos que se ha tomado, para no repetir caminos ya tomados. De esta manera se estaria aplicando de mejor manera el backtracking.

3 Ventajas y Desventajas

Algunas Ventajas

1. En teoria, debería resolver el problema, aunque en tiempo puede ser infinito

Desventajas

1. Si resuelve el problema, lo resolverá en muchísimo tiempo dado que al ser random, podria hasta revisar el mismo camino dos veces (Backtracking pesimamente implementado)
2. No existen podas
3. No esta optimizado
4. No se logro aplicar el algoritmo
5. No hay heurísticas
6. No resuelve más problemas que el 1x1

4 Conclusión

Para concluir, quiero decir que claramente que la solución a la tarea no fue abordada de la manera correcta. Tuve muchos problemas en los cuales me demoraba mucho tiempo en encontrar el error, a tal punto que hubieron cosas que nunca pude solucionar. A pesar de esto, mi solución tampoco era la optima ya que no utilizaba backtracking de forma eficiente. Hubiera sido mucho mejor si mi algoritmo guardaba los caminos ya tomados y además si utilizaba muchas podas/restricciones/heurísticas adicionales.

Para las próximas tareas tendré que dedicar más tiempo para lograr una solucion correcta.