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

IIC2133 — Estructuras de Datos y Algoritmos
2019 - 2

Tarea 3

Fecha de entrega código: Por definir
Fecha de entrega informe: No hay informe

Objetivos

- Implementar una solución eficiente para un problema de combinatoria usando Backtracking
- Diseñar e implementar optimizaciones (podas o heurísticas) para el código de backtracking.

Introducción

Fuiste contactado por alienígenas por tu asombroso trabajo descifrando los mensajes enviados por ellos y te han revelado que ya están en la tierra. Planean dominar el mundo de forma pacífica y quieren tu ayuda para hacerlo a través de las leyes humanas. Para esto postularon a su representante `Kodos` a la presidencia del mundo y te necesitan para obtener los votos necesarios para ganar las elecciones.

Figura 1: El emperador supremo del mundo: Kodos

Elecciones de la tierra

En las elecciones de la tierra existen dos candidatos: `Kodos`, el candidato alienígena y `Jack Johnson`, el candidato humano. El proceso de votación se realiza en todo el mundo a la vez dividiendo a los votantes en distintos distritos con la misma cantidad de gente. En cada distrito se elige al ganador según mayoría de votos y el ganador final de las elecciones es el candidato que tiene más distritos a su favor.

Gerrymandering es una práctica en que los políticos eligen a sus votantes y no los votantes a los políticos. Sabiendo donde viven los votantes, uno puede redibujar los distritos electorales distorsionando el resultado de las votaciones, a traves de dos técnicas:

- *Packing*: Es la idea de agrupar a la mayor cantidad de votantes del partido opuesto en un solo distrito
- *Cracking*: Es la idea de separar a los votantes del partido opuesto en varios distritos, haciendo que en cada uno de ellos sean minoría

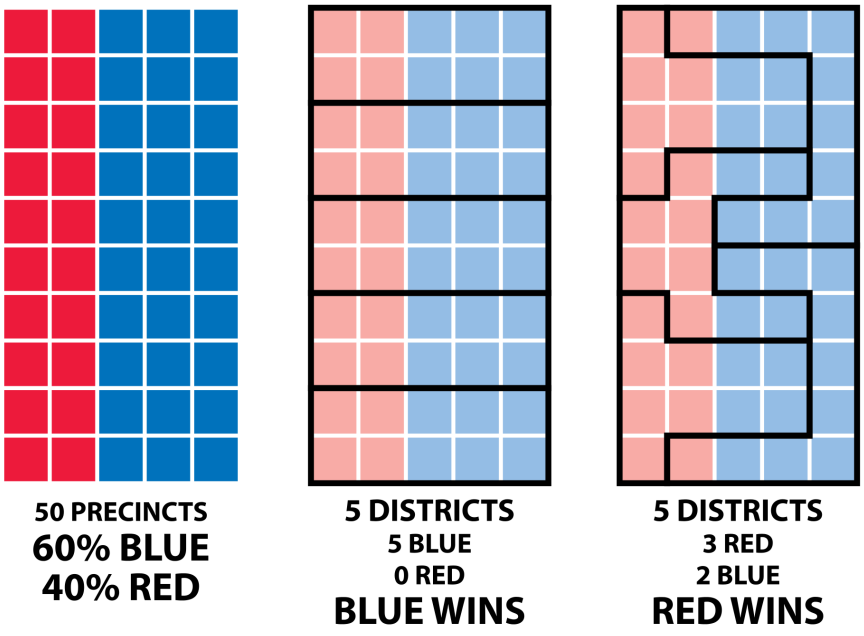

Figura 2: Ejemplo de distintos resultados electorales mediante cambios en las formas de los distritos

En la Figura 2 se ve un ejemplo del como distintas configuraciones de distritos alteran el resultado de los representantes elegidos. Teniendo un 60% de votantes azules y un 40% de votantes rojos, podemos ver que en una configuración los azules ganan el 100% de los votos y en otra el 40% de los votos.

Si miran en la Figura 2 el ejemplo de la derecha, donde gana el partido rojo se pueden apreciar las dos técnicas mencionadas. *Packing* ha sido usado en los distritos azules de más a la derecha, donde tenemos 90% de votantes azules en cada uno, y *cracking* ha sido usado en los otros, donde se tiene 40% de votantes azules en cada uno.

Definición formal del problema

Dada una grilla de $N \times M$ y un número de distritos k, asignar a cada celda de la grilla un distrito de manera de que:

- A cada celda se le asigne un distrito.
- Todas las celdas de un mismo distrito estén en contactadas entre ellas por al menos una arista (distritos contiguos).
- Todos lo distritos tengan el mismo número de **personas** (tengan cuidado: #personas != #celdas).
- El ganador de las elecciones sea `Kodos`.

Cada celda puede tomar uno de los siguientes valores:

- `EMPTY`: La celda no tiene personas en ella
- `NONE`: La celda tiene personas en ella pero no van a votar
- `PARTY_A`: La celda tiene personas y votan por KODOS
- `PARTY_B`: La celda tiene personas y votan por Jack

En la Figura 3 podemos ver un ejemplo de una grilla. En color verde podemos ver a los votantes por Kodos y en verde a los votantes por Jack. Las celdas con círculos grises representan votantes que no asisten a votar. Las celdas sin círculos son celdas sin gente, estas celdas deben ser asignadas a algún distrito y no suman votos ni número de personas a los distritos, pero pueden ser la conexión de distintas personas en un mismo distrito.

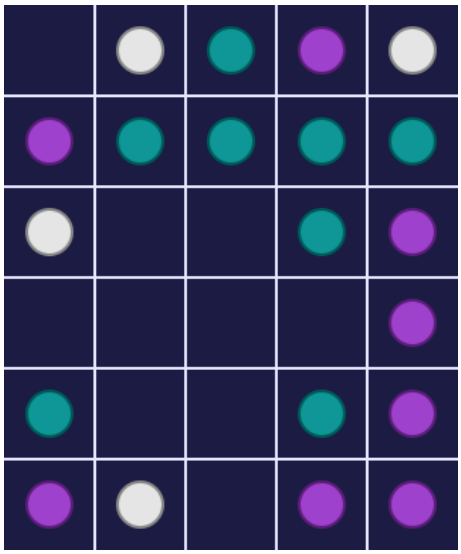

Figura 3: Grilla de ejemplo con celdas vacias, celdas con personas que no votan y votantes de Kodos y Jack

Interfaz gráfica

Para ayudarlos a debuguear, visualizar el problema y entregar el resultado, se le entrega una interfaz gráfica equipada con las funciones necesarias para hacerla funcionar. Revísala en `src/watcher/watcher.h`

Input

Tu programa se debe compilar con el comando `make` y debe generar un ejecutable de nombre `solver` que se ejecuta con `./solver test.txt` donde `text.txt` es el archivo de test que se le pasa al programa.

El archivo `test.txt` tiene el siguiente formato:

- Primera linea: El alto (N) y ancho (M) de la grilla separados por un espacio
- Segunda linea: El número de distritos (k)
- Las siguientes N lineas tendrán M dígitos cada una separados por un espacio con los siguientes valores:
  - **0** significa que la celda está vacía
  - **1** significa que la celda tiene personas que no votan
  - **2** significa que la celda tiene personas que votan Kodos
  - **3** significa que la celda tiene personas que votan Jack

Output

Como se mencionó antes, deberán trabajar con la interfaz provista. El output del programa **debe** ser mostrado en la interfaz gráfica provista por los ayudantes en el cual se muestra la asignación de los distintos distritos. Al momento de corregir, los ayudantes cambiarán el código de la interfaz gráfica para que en vez de mostrar una ventana, se escriba en un archivo, por lo tanto es importante que uses la interfaz gráfica correctamente. Los comportamientos de las funciones de la interfaz al momento de corregir serán:

- `watcher_open(int width, int height, int max_districts)`: Crea una matriz M con las dimensiones dadas y con $max\_districts$ distritos.
- `watcher_set_cell_distric(int row, int col, int district_id)`: Asigna $M[row][col]$ =district, donde 0 equivale a sin distrito y distinto de 0 es el número del distrito.
- `watcher_close()`: Guarda la matriz M en un archivo y libera la matriz. Este archivo es el que luego será corregido.
- El resto de las funciones del watcher simplemente no harán nada al momento de corregir ya que son solamente herramientas para ayudarte a visualiza la ejecución del programa.

Evaluación

Para evaluar la tarea se cambiará la interfaz gráfica por el programa descrito en la sección anterior y se ejecutará el código con `tests` de distintas dificultades con un tiempo límite de 10 segundos. Se espera que el código pueda pasar todos los `tests` (obteniendo un 7.0) sin necesidad de usar podas o heurísticas pero haciendo una modelación relativamente eficiente del problema.
Los **tests bonus** requerirán de una modelación más eficiente y uso de podas y/o herísticas del problema. Aquellos alumnos que logren pasar los `tests bonus` en menos de 10 segundos pueden sumar puntos bonus hasta tener una máxima nota de 8.0.

En esta tarea no hay informe.

Entrega

Código: GIT - Repositorio asignado. Si se crean varios repositorios cuando accedes al link para crear repositorios, trabaja en el que **no tenga** un número al final y borra el resto de los repositorios. Se entrega a mas tardar el día de entrega a las 23:59 hora de chile continental.