Trabajo práctico 3

Fecha de reentrega: 10 de diciembre, hasta las 23:59.

Fecha de entrega: 9 de noviembre, hasta las 23:59 horas.

1. Condiciones de entrega y términos de aprobación

Este trabajo práctico consta de varias secciones que deben ser resueltas y para aprobar el trabajo se requiere aprobar todas las secciones. El trabajo práctico deberá resolverse en grupos de 4 personas. De ser necesario (o si el grupo lo desea) el trabajo podrá reentregarse una vez corregido por los docentes y en ese caso la reentrega deberá estar acompañada por un *informe de modificaciones*. Este informe deberá detallar brevemente las diferencias entre las dos entregas, especificando los cambios agregados y/o partes eliminadas del trabajo. Cualquier cambio que no se encuentre en dicho informe podrá no ser tenido en cuenta en la corrección de la reentrega.

Para cada sección donde se pide encontrar una solución algorítmica al problema propuesto, desarrollar los siguientes puntos:

1. Explicar de forma clara, sencilla, estructurada y concisa, las ideas desarrolladas para la resolución del problema. Para esto se sugiere utilizar pseudocódigo y lenguaje coloquial combinando adecuadamente ambas herramientas. Es importante que lo expuesto en este punto sea suficiente para el desarrollo de los puntos subsiguientes, pero no excesivo (**¡no es un código fuente!**).
2. Deducir una cota de complejidad temporal del algoritmo propuesto en función de los parámetros que se consideren correctos y justificar por qué el algoritmo desarrollado para la resolución del problema cumple la cota dada.
3. Dar un código fuente claro que implemente la solución propuesta. El mismo no sólo debe ser correcto sino que además debe seguir las *buenas prácticas de la programación* (nombres de variables apropiados, estilo de indentación coherente, modularización adecuada, etc.). Se deben incluir las partes relevantes del código como apéndice del informe entregado.
4. Realizar una experimentación computacional para medir la performance del programa implementado así como sus distintas propiedades. Para ello se deben preparar distintos conjuntos de casos que permitan analizar distintos aspectos generales como los tiempos de ejecución en función de los parámetros, las soluciones optenidas, etc. También deberán desarrollarse experimentos que permitan medir los aspectos evolutivos de las soluciones obtenidas mediante algoritmos genéticos y que permitan comparar las distintas técnicas utilizadas en el trabajo.

Respecto de las implementaciones, se acepta cualquier lenguaje que permita el cálculo de complejidades según la forma vista en la materia. Además, debe poder compilarse y ejecutarse correctamente en las máquinas de los laboratorios del Departamento de Computación. **Se debe incluir un script o Makefile que compile los distintos ejecutables para cada uno de los algoritmos implementados.**

La cátedra recomienda el uso de C++ o Java, y se sugiere consultar con los docentes la elección de otros lenguajes para la implementación. **Solo se permite utilizar 1 lenguaje para resolver todo el trabajo práctico.** Se pueden utilizar otros lenguajes para presentar resultados.

La entrada y salida de los programas **deberán hacerse por medio de la entrada y salida estándar del sistema** de forma de comunicarse correctamente mediante *pipes* con el programa *c_linea.py* provisto por la cátedra y explicado con más detalles en la sección 5.

Deberá entregarse un informe impreso que desarrolle los puntos mencionados. Por otro lado, deberá entregarse el mismo informe en formato digital acompañado de los códigos fuentes desarrollados e instrucciones de compilación, de ser necesarias. Estos archivos deberán enviarse a la dirección `algo3.dc@gmail.com` con el asunto *"TP 3: Apellido_1, ..., Apellido_n"*, donde n es la cantidad de integrantes del grupo y *Apellido_i* es el apellido del i-ésimo integrante.

2. Introducción

Los docentes del laboratorio son aficionados jugadores del juego *C en linea* y se enteraron que el campeonato mundial de dicho juego se llevará a cabo en su ciudad. Ante la noticia no dudan en inscribirse para participar, pero como quieren ganar dicho concurso, y aprovechando que están a cargo del laboratorio de *Algoritmos y estructuras de datos III*, deciden presentar un trabajo práctico cuyo único proposito ~~es el de conseguir un programa para dicha competencia~~ es hacer que los alumnos aprendan y tengan oportunidades de ganarlo. Para asegurarse de esto, y dado que hay discusiones internas sobre si los alumnos podrán o no implementarlo, deciden guiar el proceso para que se realicen distintas implementaciones y experimentaciones y en última instancia culminar en una competencia a fin de seleccionar al mejor jugador de *C en linea* implementado por los alumnos ~~para llevar a la competencia~~ para divertirse.

3. Reglas del juego *C en linea*

C en linea es una versión de conocido juego *cuatro en linea* [2]. El mismo consiste en un tablero de N columnas y M filas y dos jugadores que cuentan con p fichas cada uno, cada jugador tiene fichas de un mismo color y las fichas de un jugador son de distinto color a las fichas del otro jugador. Los jugadores se alternan en poner las fichas en el tablero. En cada turno un jugador elige una columna para colocar la ficha y esta se ubica en la última casilla desocupada (la menor fila posible). El objetivo del juego es conseguir una linea de fichas del mismo color de tamaño mayor o igual a c, el primer jugador en conseguirlo gana. La linea puede ser tanto horizontal, vertical o diagonal, pero debe ser siempre una linea recta [3]. **Nota:** *Cuatro en linea* es una instancia particular de *C en linea* donde $N = 7$, $M = 6$, $p = 21$ y $c = 4$.

4. Consignas a realizar

1. Soluciones exactas:
   - *a*) Implementar un algoritmo óptimo para jugar a *C en linea* utilizando la técnica Minimax.
   - *b*) Mejorar el algoritmo anterior mediante la técnica de poda alfa-beta.
2. Soluciones heurísticas:
   - *a*) Implementar un algoritmo de heurística golosa, para esto se pide realizar los siguientes pasos:
     1) Implementar una función parametrizable que dado un estado cualquiera de un tablero, la misma determine la próxima jugada a realizar. La complejidad de está función debe ser como máximo $\mathcal{O}(NM)$.
     2) Implementar un jugador que utilice la función del punto $2a1$ como estrategia para elegir en cada paso la jugada a realizar.
3. Optimización de los parámetros:
   - *a*) Utilizar la técnica de *grid-search* de forma de conseguir buenos valores para los parámetros de la heurística propuesta en el punto 2.
   - *b*) Utilizar la técnica de algoritmos genéticos para optimizar los parámetros de la función desarrollada en 2. Para esto definir a los cromosomas como la parametrización anterior o una codificación de la misma e implementar lo siguiente:
     1) Una población (conjunto de cromosomas).
     2) Al menos dos funciones de *fitness*
     3) Al menos una operación de *crossover*
     4) Al menos una operación de mutación
     5) Al menos dos métodos de selección distintos
4. Leer el paper [1] y realizar un análisis comparativo respecto a lo realizado en los puntos anteriores. También se pide realizar los experimentos necesarios para poder comparar los resultados de sus implementaciones con los resultados mostrados en el paper.

5. Utilizar los parámetros óptimos obtenidos en el punto 3 y dar dos jugadores (uno con los parámetros de *grid-search* y otro con los del algoritmo genético) que sepan jugar en un tablero de *Cuatro en linea*. Estos jugadores deben ser capaces de vencer a los jugadores de la cátedra de forma consistente (ganarle más del 75 % de los partidos jugando 1000 o más partidos).

6. **Bonus:** El día 24 de noviembre se llevará a cabo una competencia de *c en linea* para la cual cada grupo compite con jugadores que deberán subministrar para los valores de N, M, c y p mostrados a continuación, la modalidad de la competencia será revelada el mismo día junto con los premios.

   - $N \in [4, \ldots, 20]$.
   - $M \in [N-1, N, N+1]$
   - $p = 2 * N * M$.
   - $c \in [3, \ldots, 8]$ siempre que cumpla $c \leq \text{mín}\{N, M\}$.

5. Especificaciones de los programas a desarrollar y del entorno a utilizar

Cada jugador es representado por un programa que lee la configuración inicial y las jugadas de su rival por la entrada estándar y escribe sus propias jugadas en la salida estándar.

La cátedra provee un programa que hace de juez (c_linea.py). Este programa puede ser provisto de hasta dos binarios y se encarga de ejecutarlos, comunicarles las jugadas del otro jugador, recibir sus jugadas y coordinar el partido, además, cuenta con la opción de mostrar mediante una interfaz gráfica el estado del juego y mediante archivos de logs. También provee algunos jugadores sin una estrategia inteligente al solo efecto de facilitar las pruebas y la posibilidad de jugar manualmente contra otra persona, contra uno de los jugadores provistos o contra un binario pasado por parámetro.

La primera línea de la entrada estándar leída por un jugador en cada partido será la configuración inicial en forma de cuatro enteros N, M, p y c separados por espacios. N es la cantidad de columnas, M es la cantidad de filas, p es la cantidad de fichas de cada jugador y c es el tamaño de la linea a formar. Luego sigue una linea que le indica al jugador si le toca empezar o no (*vos*, *el* respectivamente).

A partir de ahí, cada línea leída de la entrada estándar será una jugada del rival y cada línea escrita en la salida estándar será una jugada propia. Cuando el partido termine, el juez enviará algunos de los mensajes *perdiste*, *ganaste* o *empataron* indicando que el partido terminó y el resultado del mismo.

Luego de recibir el resultado de un partido, automaticamente empieza otro partido, esto se repite tantas veces como iteraciones fueron especificadas por parámetro al juez (si este parámetro no está especificado, se juegan infinitos partidos). En caso de que al menos uno de los jugadores sea un humano, cuando un partido termina, hay que hacer click en la pantalla para pasar al siguiente.

Cuando el juez decide terminar, ya sea por que se cumplieron las iteraciones a realizar o porque manualmente se cerró el programa mediante la cruz con el mouse o mediante la tecla *Esc* (en caso de que al menos un jugador sea humano), se envía el mensaje *salir* a los jugadores para que los mismos puedan correctamente terminar sus ejecuciones y cerrar sus recursos (en caso de tenerlos). Una vez realizado esto y antes de terminar, los jugadores deben responder *'listo'* al juez indicando que el también puede terminar.

Nota: Tener cuidado con los mensajes ya que los mismos son *case-sensitive*

Los parámetros del programa c_linea.py están explicados en el mismo programa utilizando el flag –help para visualizar la ayuda donde está detallado como utilizar los parámetros disponibles junto con ejemplos ilustrativos de como correrlo.

5.1. Ejemplos de como correr el juez

Dos jugadores humanos jugando al c_linea con la configuraciones de cuatro en linea:

```
python c_linea.py --first azul --ui True --columns 7 --rows 6 --p 21 --c 4
```

Jugador rojo humano y jugador azul bot random de la cátedra jugando al c_linea con la configuraciones de cuatro en linea:

```
python c_linea.py --blue_player ./random_player --first azul
                  --ui True --columns 7 --rows 6 --p 21 --c 4
```

5.2. Ejemplo de input/output

A continuación se muestra un ejemplo de una entrada y su posible salida. Dado que la entrada y salida corresponde a un protocolo de comunicación entre el juez y un jugador, los mismos se muestran de forma intercalada para expresar la temporalidad en que cada proceso envía sus mensajes.

Entrada de ejemplo	**Posible salida esperada para la entrada de ejemplo**
rojo azul 1 2 4 4 3 21 vos 0 0 ganaste rojo azul 1 2 4 4 3 21 vos 1 3 2 perdiste salir	 2 3 1 3 3 0 3 listo

Referencias

[1] Gregor Hochmuth. On the genetic evolution of a perfect tic-tac-toe strategy. *Genetic Algorithms and Genetic Programming at Stanford*, pages 75–82, 2003.

[2] Wikipedia. Conecta 4 — wikipedia, la enciclopedia libre, 2017.

[3] Wikipedia. Recta — wikipedia, la enciclopedia libre, 2017.