Proyecto final - Problema CalDep
Análisis y diseño de algoritmos I

1st Cristián David Zúñiga Gutierrez
202259425 - 2724

2nd Alejandro Marín Hoyos
202259353 - 3743

3rd Juan José Marín Arias
202259337 - 3743

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introducción

This document is a model and instructions for LaTeX. Please observe the conference page limits.

II. Definición del problema

A. Maintaining the Integrity of the Specifications

The IEEEtran class file is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

III. Solución ingenua

A. función ”backtrack”

La función **”backtrack”** implementa un algoritmo de backtracking para encontrar combinaciones óptimas de partidos. Esta función esta diseñada para explorar diferentes configuaciones de partidos, calculando costos de viaje y manteniendo un registro de los equipos que han jugado.
Su funcionamiento consta de un **caso base** donde si ”n” es $\mathcal{O}$, significa que se ha llegado a una configuración completa, luego se verifica si esta solución es valida y de ser así se agrega a ”rs”. Si aún quedan decisiones por tomar ($”n” > \mathbf{O}$), la funcion itera sobre cada equipo ”team” y explora las posibles combinaciones de partidos, llamandose recursivamente para este fin. Cuando se toma una decisión (por ejempllo, programar un partido entre dos equipos), se actualizan varios estados como: ”played”, `"current_post"`, `"current_post"`, ”travel” y se modifica `"remanning_matches"` para reflejar esta decisión. Después de explorar una rama de decisiones, la función ”retrocede”, deshaciendo las últimas decisiones y restaurando el estado anterior para explorar diferentes ramas de decisiones. Esto se logra mediante la reversión de los cambios en las variables de estado y la eliminación de las últimas decisiones de ”tmp”, finalmente una vez que se han explorado todas las posibilidades, la función retorna ”rs”, que contiene todas las soluciones válidas encontradas.

B. función `"find_all"`

La función **`"find_all"`** se encarga de encontrar la solución óptima para el calendaario, llamando a **”backtrack”** para cada semana del calendario y manteniendo un registro del costo total óptimo y la solución óptima encontrada hasta el momento. Recibe como parámetros un ”N” que es el número total de equipos, un ”K” que es el contador de las semanas del calenadrio, ”res” que es una lista para almacenar las combinaciones de partidos y ”tmp” que es una lista temporal para lamacenar información mientras se exploran combinaciones de partidos.
La función comienza verificando si ha alcanzado el final de la programación (todas las semanas han sido programadas). Si se ha llegado al final y el costo reciente de la configuración `"f_recent"` es mejor que el óptimo encontrado hasta el momento `"f_opt"`, se actualiza la solución óptima. De lo contrario la función incrementa el contador de semanas ”k” y llama a la función ”backtrack” para encontrar todas las posibles configuraciones de partidos para la semana actual, este proceso se realiza recursivamente para cada semana hasta que todas las semanas han sido programadas. Dentro de cada llamada a ”backtrack”, se explora el espacio de soluciones para esa semana en particular, probando diferentes combinaciones de partidos y calculando sus costos asociados, para cada configuración válida devuelta por ”backtrack”, la función `find_all` actualiza el costo reciente `"f_recent"` y agrega la configuración a la lista de resultados ”res”. Luego, realiza llamadas recursivas a sí misma para continuar construyendo la solución en semanas subsiguientes. Después de explorar una configuración, la función retrocede, es decir, revierte los cambios realizados en el estado del programa (como las posiciones actuales y previas de los equipos) y retira la última configuración de partidos de la lista de resultados ”res”, esto permite que la función explore otras configuraciones de partidos alternativas en las semanas siguientes, finaalmente una vez exploradas todas las configuraciones para todas las semanas, la función devuelve la mejor solución encontrada (solution) junto con su costo `"f_opt"`.

C. Análisis temporal y espacial

Para el análisis temporal tenemos que:

- La lectura del archivo tiene una complejidad de $\mathcal{O}(n^2)$ donde n es el número de equipos. Cada línea después de la primera se lee y se procesa en $\mathcal{O}$(n), y esto se repite n veces.
- La complejidad de la función ”backtrack” es de $\mathcal{O}(2^n)$ ya que la función itera sobre cada equipo y realiza backtracking para los partidos restantes.
- La función "`find_all`" tiene una complejidad tempral similar a la función ”backtrack” ya que itera sobre todas las posibles semanas (hasta 2n - 2) y llama a ”backtrack”. La complejidad es exponencial y posiblemente mayor a $\mathcal{O}(2^n)$ dado a que itera en el número de equipos y el número de semanas.
- Para la complejidad temporal de la construcción de la matriz sera de $\mathcal{O}(n*w)$ donde n es el número de equipos y W es el número de semanas, ya que se iteran todas las semanas y todos los equipos para cada semana.

Para el análisis espacial tenemos que:

- Se inicializan varias listas de tamaño n y su complejidad espacial sera de $\mathcal{O}(n)$ para cada una totalizando una complejidad de $\mathcal{O}(n)$ ya que crecen linealmente con el número de equipos y un¡ complejidad de $\mathcal{O}(2^n)$ para la matriz.
- La función ”backtrack” Utiliza espacio adicional para el conjunto ”tmp” y la recursión, el espacio utilizado depende del número de llamadas recursivas y el tamaño del conjunto ”tmp”, que puede crecer con el número de equipos. La complejidad espacial puede ser significativa, especialmente debido a la profundidad de la pila de llamadas en la recursión.
- La función "`find_all`" tiene una complejidad similar a backtrack, pero se agrega la lista ”res” para mantener las soluciones. La complejidad espacial es alta debido a la naturaleza recursiva y al almacenamiento de soluciones intermedias.
- Para la complejidad espacial del almacenamiento de la matriz sera de $\mathcal{O}(n * w)$ donde n es el número de equipos y W es el número de semanas, ya que esta matriz almacena los resultados de todos los partidos para cada semana.

Al centrarnos en las dos funciones principales vamos a encontrar que la complejidad computacional es de $\mathcal{O}(n!)$ ya que se deben explorar todas las posibles permutaciones de los n equipos y encontramos que hay factorial de n formas de ordenar los n equipos.

D. Análisis de cercanía al optimo