.UBAfiuba
FACULTAD DE INGENIERÍA

Teoría de Algoritmos
(75.29) Curso Buchwald - Genender

Trabajo Práctico 2
Programación Dinámica

10 de octubre de 2023

Stephanie Izquierdo
104196

Tomás Macía
99248

1. Análisis del problema

1.1. Problema

Luego de haber analizado todos los rivales ahora Scaloni lo que necesita es definir un cronograma de entrenamiento y desea hacerlo para los próximos n días.

Es sabido que el entrenamiento del día i demanda una cantidad de esfuerzo e_i , y podemos decir que la ganancia de dicho entrenamiento es igual al esfuerzo.

El entrenamiento que corresponde al día i (así como su esfuerzo y ganancia) son inamovibles. Si la cantidad de energía que se tiene para un día i es $j < e_i$, entonces la ganancia que se obtiene en ese caso es j.

Además se debe tener en cuenta que la cantidad de energía que tienen disponible para cada día va disminuyendo a medida que pasan los entrenamientos $s_1 > s_2 > ... > s_n$

Existe también la posibilidad de descansar Un día logrando así que la energía se renueve es decir que la energía vuelva a ser s_1

1.2. Entendiendo el problema

Habiendo explicado el problema se plantea como objetivo definir los días que deben entrenarse y los días que convenga a descansar de tal forma de obtener la mayor ganancia posible.

La problemática reside en que para obtener la mayor ganancia hay que definir cuándo nos conviene descansar de forma tal que el reinicio de energía sea bien aprovechado para el día actual ya que el esfuerzo inicial es el mayor y este va decreciendo según el pasar de los días.

Observamos entonces que la energía del día no varía, pero sí el esfuerzo gracias a este reinicio.

Entonces debemos determinar qué día descansar teniendo en cuenta las distintas posibilidades de ganancia según si anteriormente descansó el equipo.

Por ejemplo se podría considerar entrenar todos los días seguidos pero en el caso en que el esfuerzo actual sea menor al inicial y la energía del día actual sea mayor igual a la inicial convendría descansar el día anterior para poder aprovechar al máximo esa energía.

	1	2	3
Energia	5	27	13
Esfuerzo	35	14	10

Cuadro 1: Ejemplo promedio

1.3. Solución

Con lo cual nosotros planteamos los siguientes casos da las combinaciones posibles que se puede tener entre los descansos y entrenamientos:

- Como es posible entrenar o no entrenar el primer día la fórmula para los dos primeros días resulta en la siguiente: $min(e_i, s_j)$ siendo j la representación del primer día o segundo.
- Si se entrena todos los días consecutivos se obtendrá una ganancia igual a: $min(e_i, s_i)$ + ganancia del día anterior = $min(e_i, s_i) + G(e_{i-1}, s_{i-1})$
- Entendemos también que es redundante plantear esfuerzos de día futuros para el día actual
- Para el primer esfuerzo, hay que tener en cuenta que podemos obtenerlo habiendo descansado un día que es mayor a 1, con lo cual para este caso consideramos el mínimo entre el esfuerzo actual más la ganancia de el óptimo de dos días atrás Ya que esto nos indica si se tomó un

descanso o se entrenó consecutivamente: $min(s_j, e_i) + max(e_{i-2})$ Siendo J distinto de y J menor a i

Presentando así en la siguiente fórmula de recurrencia:

- Para el caso en el que se entrena el día anterior: $min(e_i, s_j) + M(e_{j-1}, s_{j-1})$
- Para el caso en que se descansa el día anterior: $min(s_j, e_i) + max(e_{i-2})$

2. Algoritmo

En la sección anterior fue descrito el algoritmo en términos de pseudocódigo y los pasos a seguir para ir obteniendo los óptimos. A continuación se presenta el código del algoritmo desarrollado.

```
def scaloni_ganancia(e_i, s_i):
  cant_dias = len(e_i)
  matriz = np.zeros((cant_dias, cant_dias), dtype=int)

  for columna in range(len(matriz[0])):
    for fila in range(len(matriz)):
      if fila == 0:
        if columna==0 or columna==1:
          matriz[fila][columna] = min(s_i[fila], e_i[columna])
        else:
          matriz[fila][columna] = min(s_i[fila], e_i[columna]) + max(matriz[:,
    columna-2])
      elif fila > columna: continue
      else:
          matriz[fila][columna] = min(s_i[fila], e_i[columna]) + matriz[fila-1,
    columna-1]
  return matriz
```

La complejidad del algoritmo propuesto para encontrar el orden optimo de entrenamiento es $\mathcal{O}(n^3)$, debido a que necesitamos generar una matriz de tamaño n x n siendo n la cantidad de dias y luego para cada posición se realizan operacion de minimo y maximo siendo la de minimo $\mathcal{O}(1)$ ya que comparamos 2 valores y la de maximo $\mathcal{O}(n)$ ya que comparamos toda una fila. Dando asi una complejidad de $(n^2 * n)$

Pero además como de la matriz resulta triangular debido a las aclaraciones anteriores obtenemos como resultado $(n^2 * n/2)$ Lo cual resulta en $\mathcal{O}(n^3)$

El algoritmo para obtener la ganancia máxima obtenida por la construcción de la matriz anterior es el siguiente:

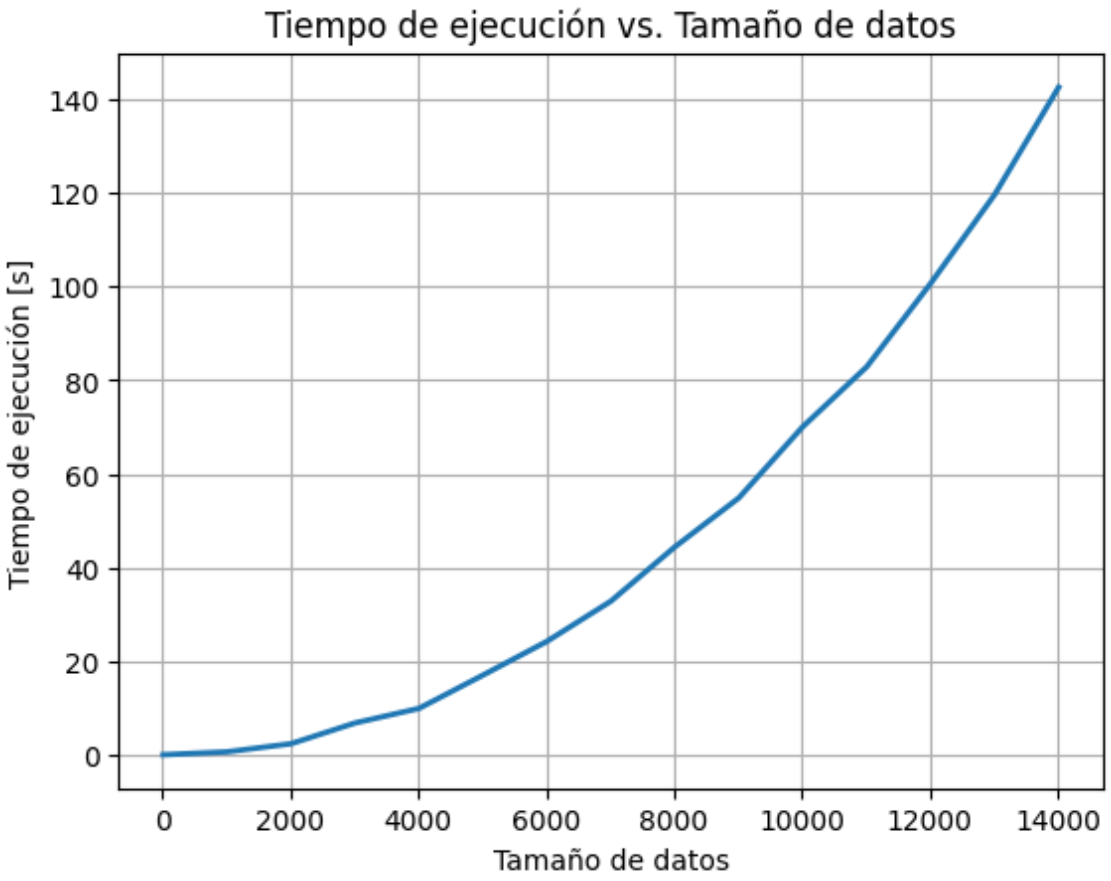
```
def scaloni_plan(opt):
  cant_dias = opt.shape[1]
  idx_fila, idx_col = (np.argmax(opt[:, -1]), cant_dias - 1)
  plan = [ENTRENO]

  while len(plan) != cant_dias:
    if idx_fila != 0:
      idx_fila -= 1
      idx_col -= 1
      plan.append(ENTRENO)
    else:
      idx_fila = np.argmax(opt[:, idx_col-1]) # check if idx_col == 0
      idx_col -= 1
      plan.append(DESCANSO)

  return plan[::-1]
```

3. Mediciones

Se realizaron mediciones en base a crear distintas combinaciones de e_i y s_i, cuyos valores se encuentran entre 1 y 100. Se ejecutó el algoritmo con distintos valores de n, es decir el tamaño de los datos (días de entrenamiento). Estos datos van desde 10 elementos hasta aproximadamente 15000 elementos, con un intervalo de 100. Es decir se calculó con 10, 1010, 2010.... 14010 elementos y se midió el tiempo que tardó únicamente la ejecución del algoritmo (el tiempo de generación del set de pruebas random no fue medido). La elección de estos valores y saltos fueron elegidos en base a prueba y error y a los tiempos que nos demoró en realizar ejecuciones con distintos tamaños de datos



4. Conclusiones

En síntesis, el enfoque desarrollado en este trabajo aborda eficazmente el desafío de optimizar la ganancia de análisis de dias de entrenamiento del equipo de Lionel Scaloni. Se ha identificado que el factor crítico es reducir al máximo el tiempo de espera de los ayudantes que más demoran en terminar su estudio del rival, para comenzar cuanto antes su tarea de análisis y así terminar lo antes posible. A través de un algoritmo programación dinamica, se logra asignar la ganancia de manera estratégica teniendo en cuenta el reinicio de esfuerzo. Esta solución ofrece una forma efectiva y eficiente de abordar el problema, permitiendo a Scaloni entreanar de manera optima a su equipo.