UNIVERSIDADE DE ÉVORA

Departamento de Informática - Universidade de Évora

Trabalho de EDA2 2023/24

" Assignment 1: Board Game Party "

Diogo Matos Aluno 54466
Ivo Rego Aluno 32640

2 de abril de 2024

1 Introdução

- A otimização de seleção de grupos de jogos usando programação dinâmica, apresenta um desafio na area da programação, desde a alocação de recursos até a maximização de resultados em condições limitadas. O trabalho prático em questão, analisa uma solução de programação para o problema de selecionar grupos de jogos para maximizar o entusiasmo total, dado um limite de espaço disponível, representado pelo número de jogadores que podem ser colocados. O problema é resolvido utilizando conceitos fundamentais de Estruturas de Dados e Algoritmos, com foco especial na programação dinâmica.

2 Metodologia

- Para aplicação de conceitos da disciplina de Estruturas de Dados e Algoritmos, a resolução do problema proposto, é implementado em Java. O programa utiliza uma combinação de classes, listas e arrays bidimensionais para representar grupos de jogos e calcular a seleção ótima.

3 Descrição do Algoritmo

- O programa desenvolvido usa a técnica de Programação Dinâmica para resolver o problema de selecionar um conjunto ótimo de grupos de jogos que maximiza o entusiasmo total, dado um espaço limitado para os jogadores. De realçar que o código tem bastantes comentários para ajudar a entender a implementação.

- **Passos do Algoritmo:**

  Inicialização: São criadas duas matrizes principais, dp e selected, onde dp[i][j] armazena o máximo entusiasmo que pode ser alcançado com os primeiros i grupos e com um limite de j jogadores. A matriz selected é usada para rastrear quais grupos foram escolhidos para alcançar o entusiasmo máximo em dp[i][j].

  Preenchimento das Matrizes: O algoritmo itera sobre cada grupo de jogos e cada possível quantidade de espaço de jogadores, decidindo se o grupo atual deve ser incluído no conjunto ótimo. A decisão baseia-se na comparação entre o entusiasmo alcançado incluindo o grupo atual e o entusiasmo sem incluí-lo.

  Seleção de grupos: Após preencher as matrizes, o algoritmo reconstrói o conjunto de grupos selecionados para alcançar o máximo entusiasmo, percorrendo a matriz selected de trás para frente.

  Output: Por fim, o programa imprime o número de grupos selecionados, o total de jogadores nesses grupos, e o entusiasmo total, seguido pelos nomes dos grupos selecionados.

4 Função Recursiva e Programação Dinâmica

- A base da solução é a função selectGames, que implementa o conceito de programação dinâmica para calcular o máximo entusiasmo possível. Diferente de uma abordagem puramente recursiva, a programação dinâmica armazena os resultados de subproblemas para evitar recalculos, aumentando muito a eficiência.

5 Calculo de complexidades

- **Complexidade Temporal:** O algoritmo possui uma complexidade temporal de O(N*S), onde N é o número de grupos de jogos e S é o espaço disponível para os jogadores. Isso deve-se ao fato de o algoritmo iterar através de cada grupo de jogos (N) para cada quantidade possível de espaço (S).
- **Complexidade Espacial:** A complexidade espacial é também O(N*S) devido ao armazenamento das matrizes dp e selected, que têm dimensões N+1 por S+1.

6 Conclusão e Comentários Adicionais

- Este programa é um exemplo do uso eficiente da programação dinâmica para resolver problemas de otimização. O mesmo demonstra como esta abordagem pode ser aplicada para selecionar um subconjunto ótimo de elementos (neste caso, grupos de jogos) para maximizar um determinado critério (entusiasmo), respeitando uma restrição (espaço disponível).

  A escolha por programação dinâmica é apropriada aqui, devido ao problema em questão, que apresenta uma evidente sobreposição de subproblemas e a propriedade de subestrutura ótima, características essenciais para aplicar esta técnica.

  Para ilustração e melhor compreensão, seria útil visualizar este algoritmo através de um exemplo concreto com pequenas quantidades de N e S, criando um diagrama para mostrar como os valores são calculados e armazenados nas matrizes dp e selected.

  Considerarando um exemplo simplificado para ilustrar como o algoritmo de programação dinâmica funciona, passo a passo, utilizando o programa de seleção de grupos de jogos.

  Suponhamos que temos um espaço disponível para 4 jogadores

  (S = 4) e 3 grupos de jogos (N = 3), com os seguintes detalhes:

  Grupo A: 2 jogadores, 3 de entusiasmo

  Grupo B: 3 jogadores, 4 de entusiasmo

  Grupo C: 1 jogador, 2 de entusiasmo

- **Passo 1: Inicialização das Matrizes** Inicializamos duas matrizes, dp e selected,

  ambas de tamanho [N+1][S+1] = [4][5],

  onde N é o número de grupos e S o espaço disponível para jogadores.

  Cada célula dp[i][j] representa o máximo entusiasmo possível com os primeiros i grupos e j espaço para jogadores.

- **Passo 2: Preenchimento das Matrizes** Iteramos sobre cada grupo (i) e cada espaço possível (j), decidindo se incluímos ou não o grupo atual no conjunto ótimo.

  Iteração 1 (Grupo A): Para j = 1 (espaço para 1 jogador), não podemos incluir o grupo A (precisa de 2 jogadores), então dp[1][1] = 0. Para j = 2, podemos incluir o grupo A, então dp[1][2] = 3 (entusiasmo do grupo A).

  Repetindo a lógica, dp[1][3] = 3 e dp[1][4] = 3, pois o grupo A é o único considerado até agora.

  Iteração 2 (Grupo B): Para j = 3, agora podemos incluir o grupo B ao invés de A, então dp[2][3] = 4. Para j = 4, temos uma escolha: ou incluímos A e C (3+2=5 de entusiasmo) ou apenas B (4 de entusiasmo).

  Escolhemos A e C para maximizar o entusiasmo.

  Iteração 3 (Grupo C): A inclusão de C é avaliada com os grupos anteriores, mas como C já foi considerado na escolha ótima anterior, as decisões finais não são alteradas para j=4.

- **Passo 3: Reconstrução do Conjunto Ótimo** Para reconstruir o conjunto, começamos do dp[3][4] e olhamos para as seleções feitas:

  Para j = 4, o máximo entusiasmo é 5, com os grupos A e C selecionados. Seguimos a matriz selected para identificar esses grupos.

- **Passo 4: Output** Imprimimos o número de grupos selecionados (2), o total de jogadores (3), o entusiasmo total (5), e os nomes dos grupos selecionados (A, C).

- **Visualização do Processo** Embora uma visualização gráfica específica não seja viável aqui, podemos imaginar as matrizes dp e selected sendo preenchidas ao longo dessas iterações, com as decisões de inclusão baseadas na maximização do entusiasmo sob a restrição do número de jogadores.

  Cada célula dp[i][j] reflete o melhor resultado até aquele ponto, e a matriz selected ajuda a rastrear quais grupos contribuem para esse resultado ótimo.

- **Decorrer da implementação e problemas encontrados** Durante o processo de desenvolvimento até alcançarmos a implementação final, enfrentamos diversas dificuldades. Testamos várias abordagens, algumas das quais não foram devidamente ponderadas. Muitas vezes, esquecemos certos detalhes do que foi solicitado, resultando em testes do enunciado que passavam, mas falhando em testes mais abrangentes que realizamos para aprofundar a validação do código. Essa situação obrigou nos a revisar e reformular o código repetidamente. Chegamos até a considerar uma solução recursiva, contudo, sua complexidade acabou por torná-la inviável para a entrega final.