Zajęcia E-2: “Przyśpieszanie algorytmów dynamicznych”

Cel zajęć i efekty uczenia

Główne cele zajęć / materiał do opanowania:

- Algorytm najdłuższego podciągu rosnącego
- Przykłady podejścia dynamicznego - szukanie podzadań

Zadania do rozwiązania na sprawdzarce

Remont autostrady
W autostradzie jest n dziur, na pozycjach a_1, a_2, …, a_n od początku autostrady. Naprawienie i-tej dziury kosztuje c_i. Można też przykładać długie łaty - każda łata kosztuje M i przykrywa wszystkie dziury na pewnym przedziale domkniętym długości d. Jaki jest minimalny koszt załatania wszystkich dziur?

Chorąży
Dany jest ciąg n liczb. Ile operacji przeniesienia elementu w inne miejsce trzeba wykonać, aby ciąg był posortowany?

Plan zajęć

Szacunkowy czas trwania: 2 godziny lekcyjne.

1. Zadanie “Remont autostrady”
    - Podzadanie jest analogiczne jak w poprzednio rozważanych problemach - koszt W[k] załatania dziur a_1, a_2, …, a_k.
    - *Albo dziurę a_k naprawiamy pojedynczo - wtedy koszt to $W[k-1]+c_k$, albo przystawiamy do niej łatę, przykrywając również część poprzednich dziur - wtedy koszt to $M + W[s]$, gdzie s jest największe takie, że $a_s < a_k$ - d.*
    - *Dziurę s możemy wyszukać binarnie, osiągając koszt n log n całego algorytmu.*
2. Najdłuższy podciąg rosnący - algorytm kwadratowy
    - Dla danego ciągu liczb szukamy nadłuższego podciągu (wyboru pewnych elementów ciągu, niekoniecznie sąsiednich), który jest silnie rosnący.
    - *Najbardziej intuicyjny algorytm dynamiczny: dla każdego elementu obliczamy długość podciągu rosnącego, który się na nim kończy - jeśli A[1..n] to tablica wejściowa, a W[1..n] - tablica wyników, to $W[k] = min \{ W[i] : i < k, A[i] < A[k] \}$.*
    - *Algorytm działa w czasie kwadratowym, choć można go przyśpieszyć do O(n log n) wykorzystując np. statyczne drzewa przedziałowe - oczywiście w tym momencie kursu uczniowie jeszcze ich nie znają. Tematem lekcji jest zupełnie inny, szybki algorytm na najdłuższy podciąg rosnący, opisany poniżej.*
3. Zmiana wymiaru - algorytm O(n log n)

- *Odwracamy pytanie z poprzedniego algorytmu: dla każdej możliwej długości podciągu rosnącego trzymamy minimalny element, na którym taki podciąg może się skończyć. Pełny opis i kod można znaleźć np. w tym miejscu.*