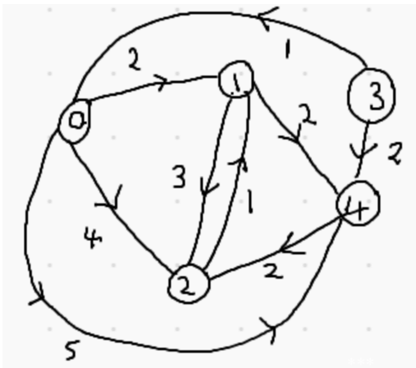
Informatics 2 – Introduction to Algorithms and Data Structures

Tutorial 6 - Greedy and Dynamic Programming

Mary Cryan

week 3: 27th to 31st January, 2025

1. In this question, we ask you to execute Dijkstra's Algorithm from node 0 on the graph, showing the steps/updates to the d and π arrays.



2. In this question we consider the *fractional knapsack problem.* We are given as input a set of n infinitely divisible items with values $v_i \in \mathbb{N}$ and sizes $w_i \in \mathbb{N}$ for $i = 1, \ldots, n$, as well as some capacity $C \in \mathbb{N}$.

   For fractional knapsack, our goal is to find optimal $x_i \in [0, 1], i = 1, \ldots, n$ for

$$\max \sum_{i=1}^{n} x_i \cdot v_i \tag{1}$$

$$\text{subject to } \; x_i \in [0,1], i = 1, \ldots, n \; \text{ and } \; \sum_{i=1}^{n} x_i \cdot w_i \leq C \tag{2}$$

   The possibility to set the x_i to any values in the $[0, 1]$ interval is what allows the items to be finely divisible (we can take any fractional amount of each item). Our goal is to have the largest possible knapsack in terms of total value (as measured in (1)).

   A greedy algorithm for the fractional knapsack problem in each step chooses an item i based on *some greedy criterion* and adds the largest possible fraction x_i of item i that is possible to fit in the (current) leftover capacity of the knapsack, without violating the overall weight constraint. The algorithm ends when the total weight of items included in the knapsack exactly meets the weight constraint, or (if C is very large) when all of the items have entirely been added ($x_i = 1$ for all i) in the knapsack. Here are 2 criteria we might use to select the next item i to be added to the knapsack:

(a) Add the item with the largest v_i value among all remaining items.
(b) Add the item with the largest v_i/w_i ratio among all remaining items.

We have two tasks for you in this question:

(i) Prove (perhaps by giving a counterexample) that the greedy algorithm with Criterion (a) does not give an optimal solution.
(ii) Prove that the greedy algorithm with Criterion (b) is optimal.

3. In this question we consider a different variant of knapsack, which is *0/1 knapsack*. This version of the problem takes exactly the same inputs as for fractional knapsack in question 2. However, in the binary scenario, we are required to "take or not-take" a specific item i, and no longer have the option to set x_i to a fractional value. For *0/1 knapsack*, every x_i must either be 0 or 1. This is the only change to (2) from Question 2, and (1) stays exactly the same.

We will exploit *dynamic programming* to solve *maximum knapsack* in the binary setting. The dynamic programming algorithm operates on a $(n+1)\cdot(C+1)$ sized array *kp*, and elies on the following recurrence:

$$kp(k+1, C') \;=\; \begin{cases} kp(k, C') & w_{k+1} > C' \\ \max\{kp(k, C'),\; v_{k+1} + kp(k, C' - w_{k+1})\} & \text{otherwise} \end{cases}$$

The base cases are specified in lines 1., 2. of maxKnapsack below.

Formally, $kp(k, C')$ is the greatest total value of a knapsack of weight $\leq C'$ that can be achieved using items $w_1, \ldots, w_k$.

In the Algorithm below, we can see the use of the recurrence in lines 5-8.

Algorithm maxKnapsack($w_1, \ldots, w_n$, C)

1. initialise row 0 of kp to "all-0s"
2. initialise column 0 of kp to "all-0s"
3. **for** ($i \leftarrow 1$ **to** n) **do**
4.   **for** ($C' \leftarrow 1$ **to** C) **do**
5.   **if** ($w_i > C'$) **then**
6. $kp[i, C'] \leftarrow kp[i-1, C']$
7.   **else**
8. $kp[i, C'] \leftarrow \max\{kp[i-1, C'], kp[i-1, C' - w_i] + v_i\}$
9. **return** $kp[n, C]$

(a) Give an $\Theta(\cdot)$ bound for the worst-case running-time of maxKnapsack in terms of n and C.
(b) Suppose the input to maxKnapsack is the list of weights $w_1 = 3, w_2 = 2, w_3 = 3$, of values $v_1 = 2, v_2 = 3, v_3 = 4$ and the capacity $C = 7$. Draw the 4×8-dimensional table kp that would be built by maxKnapsack.
(c) Consider the greedy algorithm with Criterion (b) from Question 2. Would this be guaranteed to solve the 0/1 knapsack problem optimally? Provide either a correctness proof or a counterexample to support your claim.

Mary Cryan, 23 January 2025