Fundamental Algorithmic Techniques V

February 8, 2026

Outline

The greedy algorithm paradigm

Best possible (greedy) choice right now, for immediate best outcome!

Requirements:

1. **greedy-choice property:**
   globally optimal solution $\Leftrightarrow$ local optimal (greedy) choices
2. **optimal substructure**

Examples where Greedy Algorithm is suboptimal

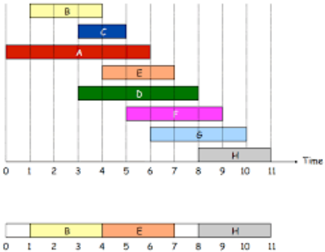
- life!?
- road...
- 0-1 knapsack problem

Examples with Greedy: Courses Allocation

Course allocation:

For starting time T:

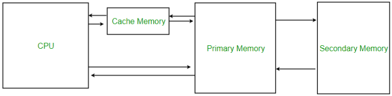
- Select out courses with starting $< \mathrm{T}$
- Choose remaining course C with lowest start time T_{end}
- Update $\mathrm{T} \leftarrow C_{T_{\mathrm{end}}}$

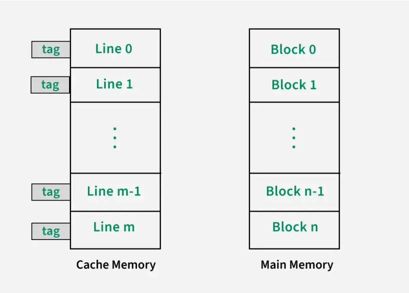


Examples with Greedy: Cache Memory Management

On request for block b_i:

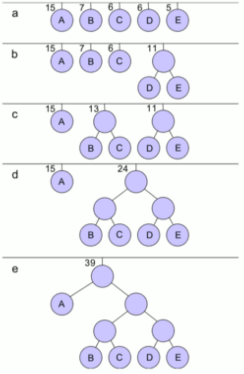
- **Hit:** b_i is in cache $\rightarrow$ no change.
- **Miss, cache not full:** add b_i.
- **Miss, cache full:** evicts one block, add b_i.

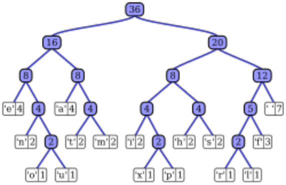




Greedy Strategy for cache allocation removing less used cache blocks

Huffman Encoding Example





Char	Freq	Code
space	7	111
a	4	010
e	4	000
f	3	1101
h	2	1010
i	2	1000
m	2	0111
n	2	0010
s	2	1011
t	2	0110
l	1	11001
o	1	00110
p	1	10011
r	1	11000
u	1	00111
x	1	10010

Huffman Code: Numerical Example

Input (A, W)	**Symbol** (a_i)					**Sum**
	a	b	c	d	e	
Weights (w_i)	0.10	0.15	0.30	0.16	0.29	$= 1$
Output C	**Codewords** (c_i)					
	010	011	11	00	10	
Codeword length (ℓ_i)	3	3	2	2	2	
$\ell_i w_i$	0.30	0.45	0.60	0.32	0.58	$L(C) = 2.25$
Optimality	**Probability budget** $(2^{-\ell_i})$					
	1/8	1/8	1/4	1/4	1/4	$= 1.00$
Info. content $(-\log_2 w_i)$	3.32	2.74	1.74	2.64	1.79	
$-w_i \log_2 w_i$	0.332	0.411	0.521	0.423	0.518	$H(A) = 2.205$

Huffman coding approximates the optimal lossless compression bound!

- The Huffman code minimizes the expected length: $L(C) = \sum_i w_i \ell_i$
- The (Shannon) entropy of the source is: $H(A) = -\sum_i w_i \log_2 w_i$
- Huffman coding is near-optimal: $H(A) \leq L(C) < H(A) + 1$

Characteristics of Greedy Algorithms

In addition to greedy property and optimal substructures...

- **A candidate set** – A solution is created from this set.
- **A selection function** – Used to choose the best candidate to be added to the solution.
- **A feasibility function** – Used to determine whether a candidate can be used to contribute to the solution.
- **An objective function** – Used to assign a value to a solution or a partial solution.
- **A solution function** – Used to indicate whether a complete solution has been reached.

Correctness Proof: Greedy Stays Ahead

Idea: Show greedy solution is *at least as far along* as optimal after each step.

- Let $A = (a_1, \ldots, a_k)$ be greedy solution, $O = (o_1, \ldots, o_m)$ optimal.
- Define a **progress measure** $\pi(\cdot)$ (e.g., finish time, coverage).
- **Key claim (by induction):** After i steps,

$$\pi(a_1, \ldots, a_i) \geq \pi(o_1, \ldots, o_i) \quad \text{(greedy is ``ahead'')}$$

- **Conclude optimality:** If greedy stays ahead for all i, then $k \geq m$. Since O is optimal, $k = m \rightarrow A$ is optimal.

Example: Interval scheduling (greedy picks earliest finish time).