Intro: Grådige algoritmer er følger en metode, hvor der bliver taget nogle valg som er lokalt optimale og håber det er globalt optimalt. Dette er dog ikke altid tilfældet, men nogle gange er det.
Altså er den forskellig fra dynamisk programmering i den forstand at der ikke tages hensyn til subproblemer når der foretages grådigt valg - Greede choice prpoerty.
Både dynamisk programmering og grådige algoritmer gør brug af optimal substruktur og derfor er der forskel på hvornår du skal bruge en dynamisk programmering løsning eller bare en grådig løsning.

Greedy strategy

Here we list the procedure one must follow in order to design a greedy algorithm. Each point will be discussed in depth under its own section.

1. **Describe** the optimization problem as one in which making a choice, we are left with one subproblem.
2. **Prove** that the greedy choice always yields a globally optimal solution.
3. **Demonstrate** optimal substructure by showing that, having made a greedy choice, what remains is a subproblem with the property that if we combine an optimal solution with the greedy greedy choice, we arrive at an optimal solution to the original problem.

Huffman codes

Snak om prefix codes med alfabet C ved fuldt binært træ. $d_T(c)$ er dybden i træet og c.freq er for en character c i C. Vis bitcost (sumtegn c:C $->$ $c.freq * d_T(c)$

Huffman algoritme (O(n lg n) bruger en min-priotets kø med freq som key. Merger de 2 mindste freq sammen til en ny node med summen af deres freq. Gøres $|C| - 1$ gang. **MED EKSEMPEL**

Lemma 16.2 med proof
Måske lemma 16.3.