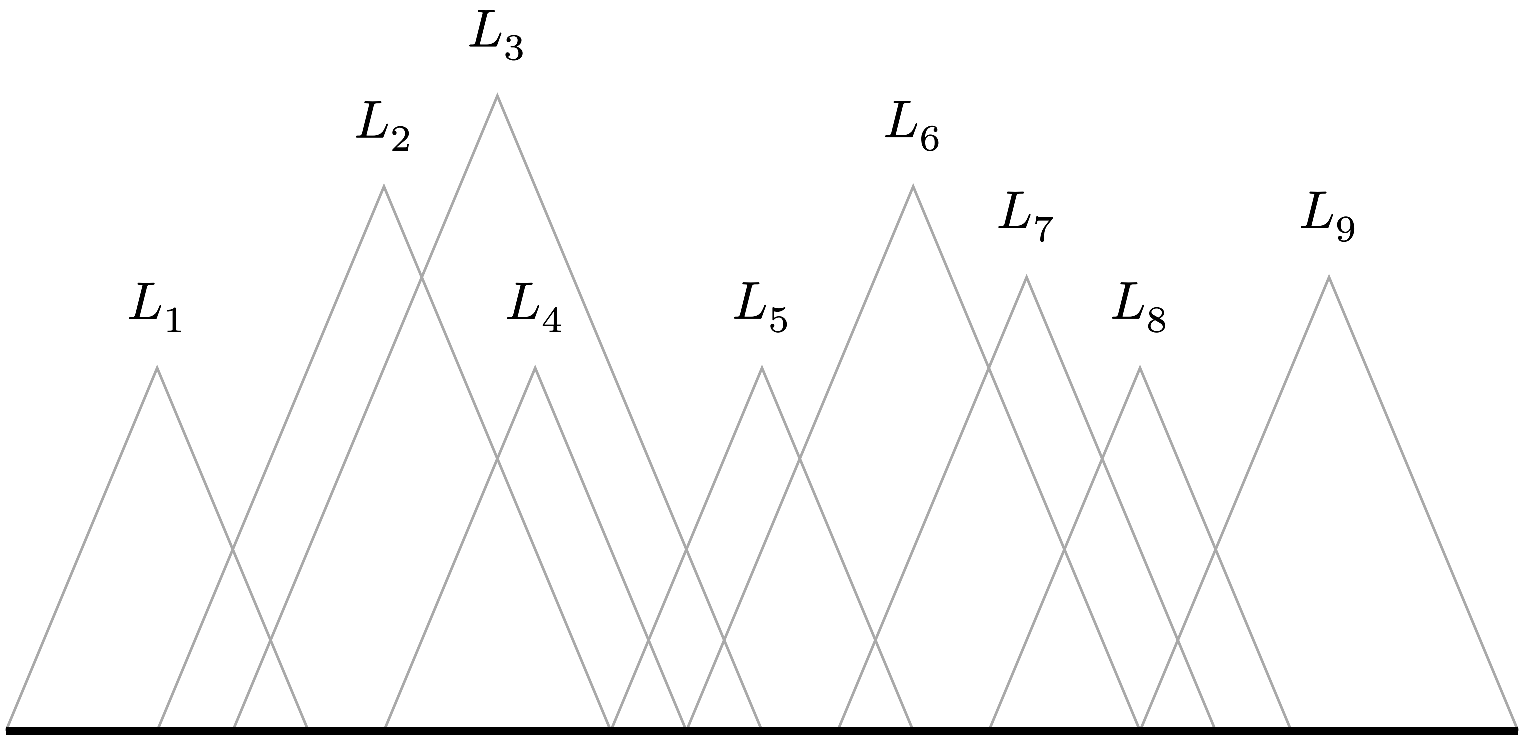
Giriga algoritmer
&
Dynamisk programmering

Exempelproblem 1

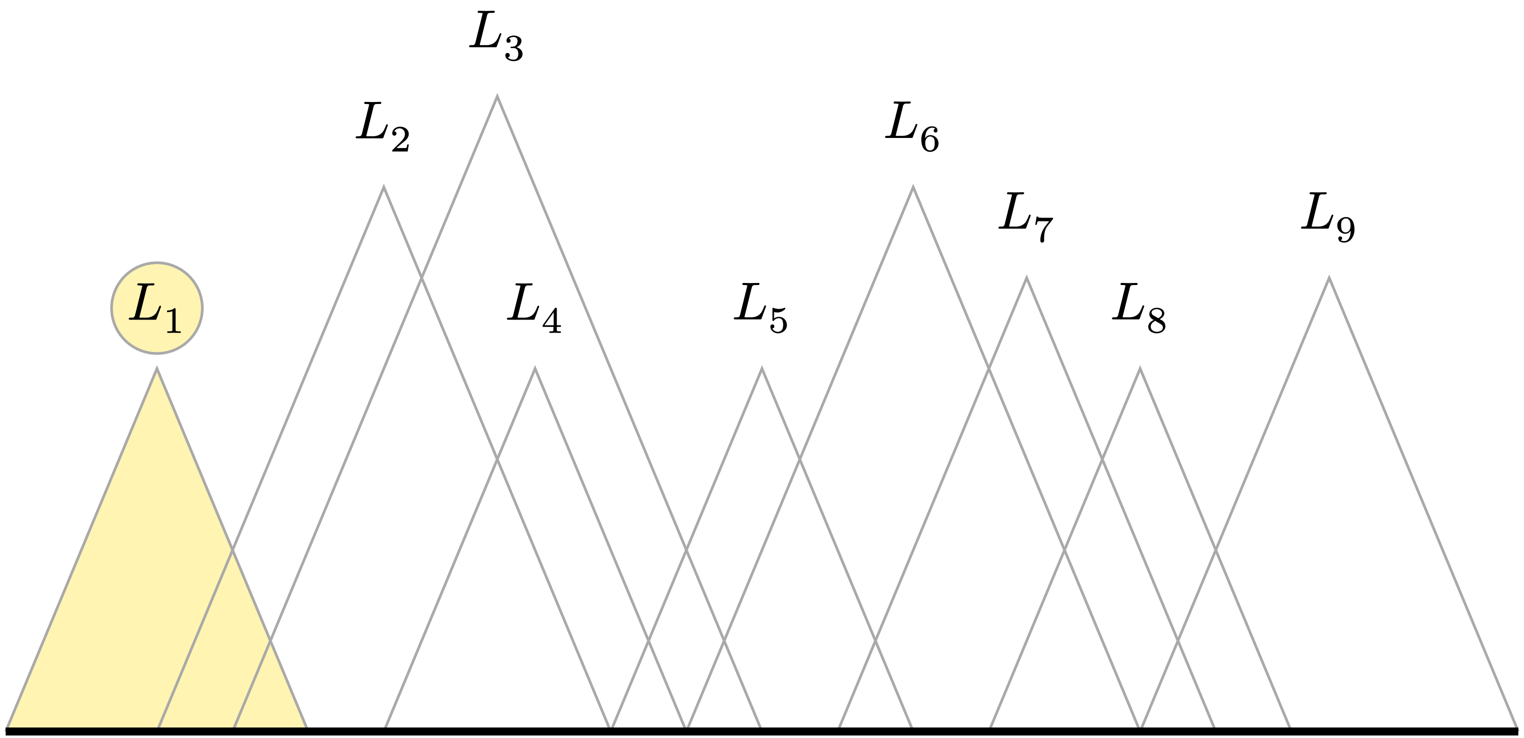
- Vi har sträcka som ska lysas upp med fasta lampor som täcker vissa delsträckor.
- Vad är det minsta antalet lampor vi behöver använda?



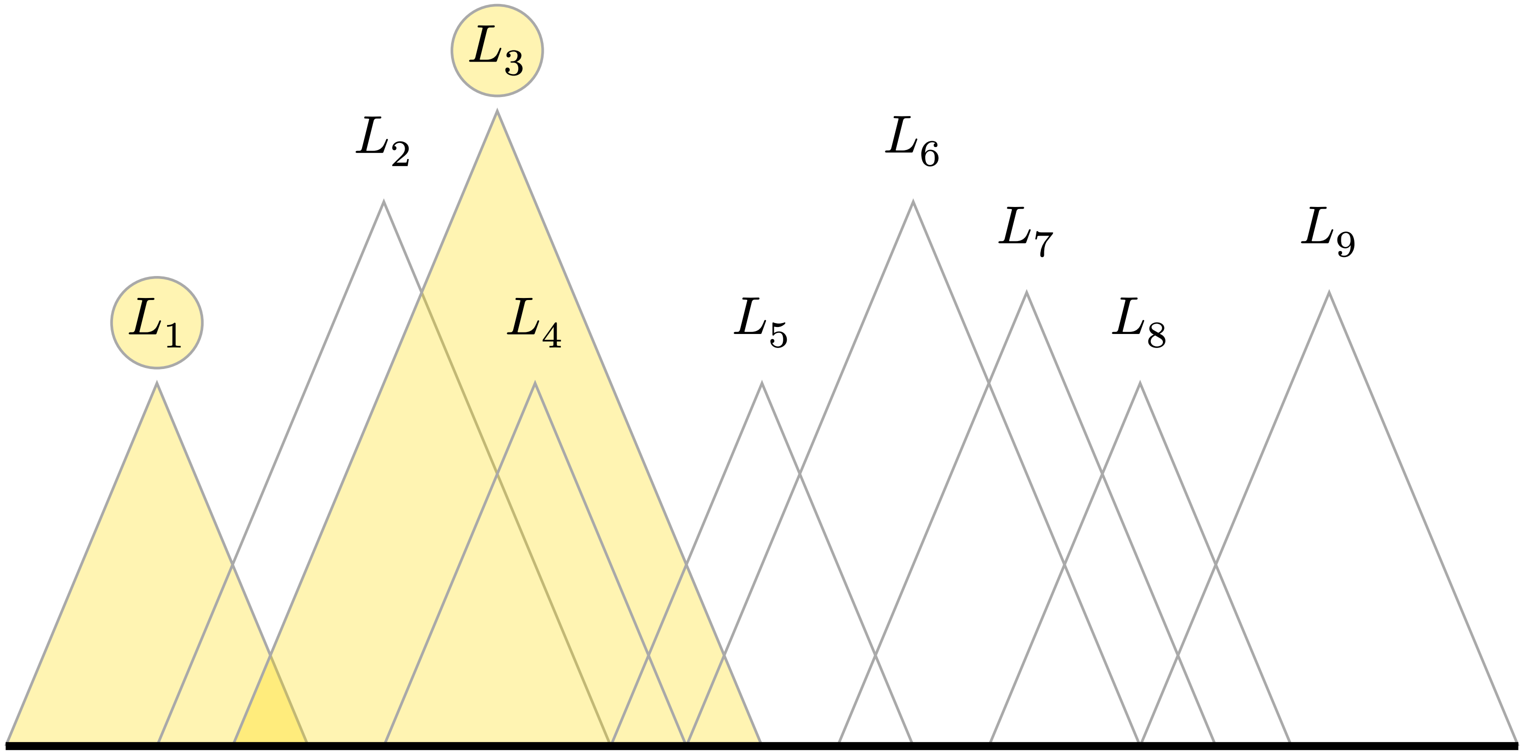
Exempelproblem 1

Svar: gå från vänster till höger och välj alltid den lampa som ökar den belysta ytan mest.

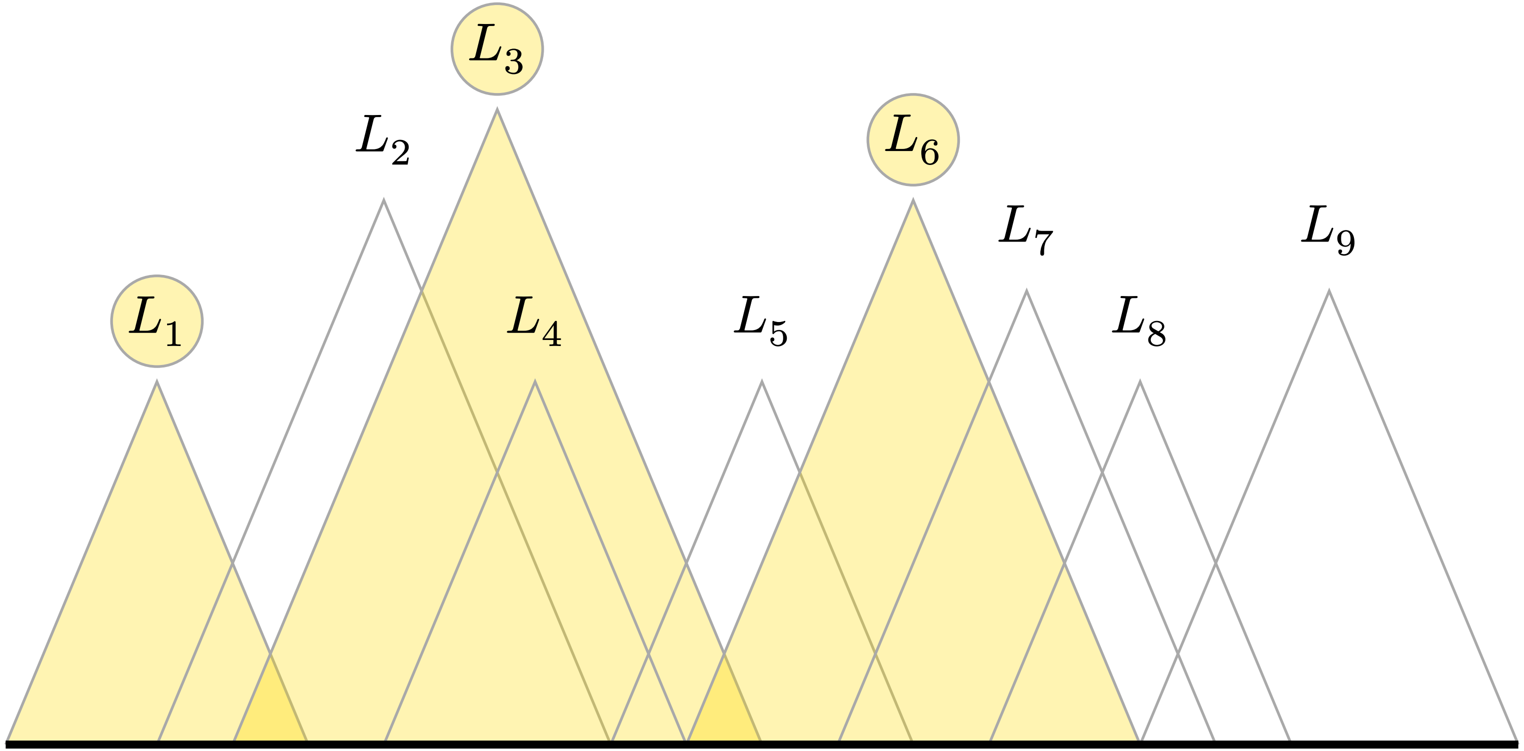
Exempelproblem 1



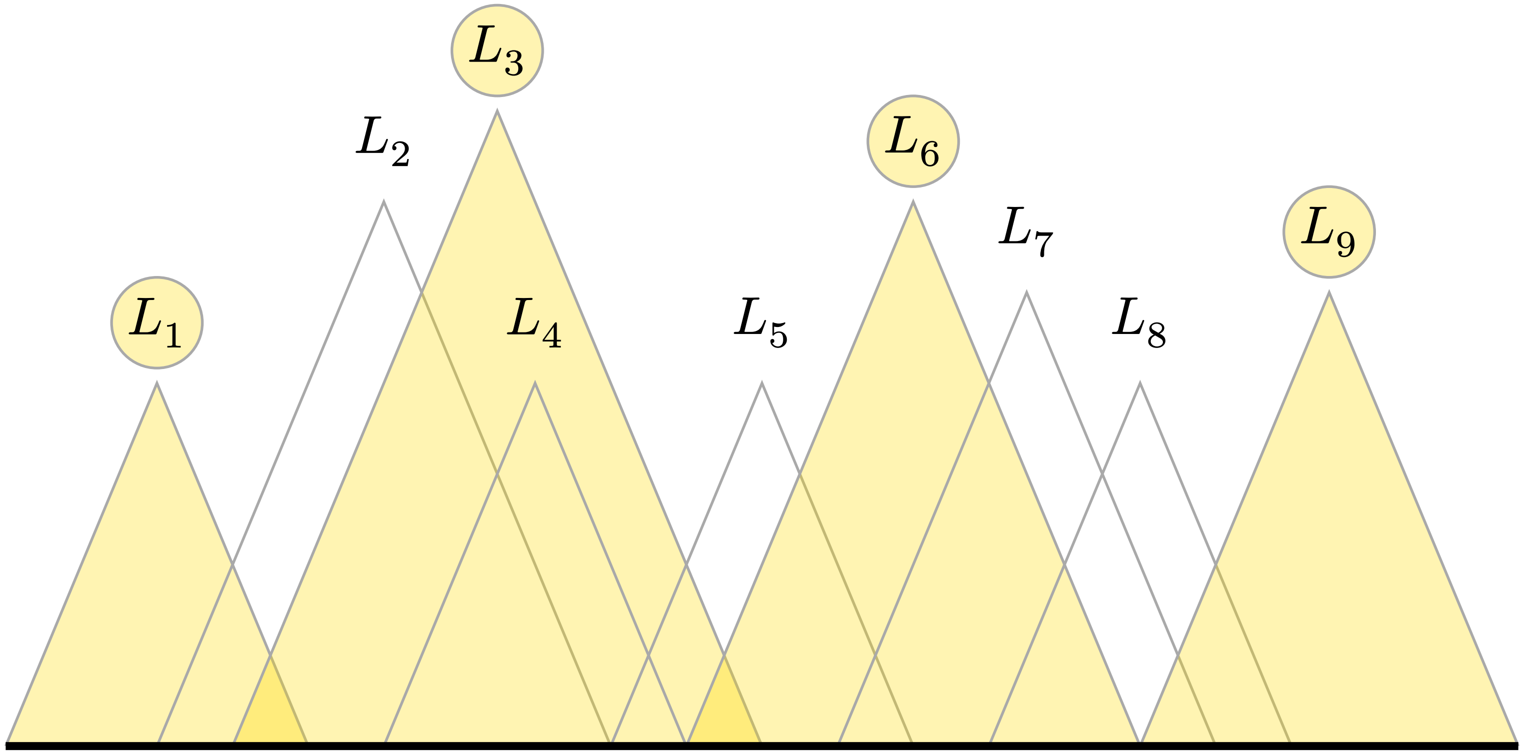
Exempelproblem 1



Exempelproblem 1



Exemplelproblem 1



Giriga algoritmer

Giriga algoritmer

- Giriga algoritmer försöker alltid ta det bästa beslutet *lokalt*
- Ingen backtracking eller hänsyn till hela indatan
- För vissa problem ger detta ett optimalt resultat

Bevisskiss för optimalitet

- Antag att vi har en girig lösning $L = \{L_1, L_2, ...L_n\}$ och en optimal lösning $O = \{O_1, O_2, ..., O_k\}$, som båda täcker intervallet $[x, y]$.
- D.v.s. $|L| = n$, $|O| = k$.

Exempelproblem 1 - bevisskiss

- Låt $\left[x_{L_1}, y_{L_1}\right] = L_1$ och $\left[x_{O_1}, y_{O_1}\right] = O_1$.
- Både L_1 och O_1 täcker x, och $y_{L_1} \geq y_{O_1}$ enligt girigheten.
- O_1 täcker $x_{O_2} \Rightarrow L_1$ täcker x_{O_2}.
- Vi kan alltså ersätta O_1 med L_1 i O.
- Vi får en ny optimal lösning $\{L_1, O_2, \ldots, O_k\}$ med längd k.

Exempelproblem 1 - bevisskiss

- Vi fortsätter tills vi har ersatt alla O_i.
- Om något $O_i \subseteq L_j$ tar vi bort O_i.
- Vi har inte skippat något L_i, så $k \leq n$.

Exemplelproblem 2

Change making problem

- Vi har en uppsättning valörer V och en summa N.
- Hur kan vi bilda summan N med så få mynt som möjligt?
- Kan vi använda en girig algoritm?

Exempel: $N = 19$, $V = [10, 5, 2, 1]$

Exemplproblem 2

Exempel: $N = 19$, $V = [10, 5, 2, 1]$
Girig lösning: använd alltid största möjliga valör:

$$
\begin{aligned}
19 - \boxed{10} &= 9 \\
9 - \boxed{5} &= 4 \\
4 - \boxed{2} &= 2 \\
2 - \boxed{2} &= 0
\end{aligned}
$$

Svar: $[10, 5, 2, 2]$ (optimalt)

Ett till exempel:

$$N = 14, V = [10, 7, 1]$$

Funkar den giriga algoritmen i detta fall?

Exemplelproblem 2

Ett till exempel: $N = 14$, $V = [10, 7, 1]$

$$\begin{aligned} 14 - \boxed{10} &= 4 \\ 4 - \boxed{1} &= 3 \\ 3 - \boxed{1} &= 2 \\ 2 - \boxed{1} &= 1 \\ 1 - \boxed{1} &= 0 \end{aligned}$$

Svar: $[10, 1, 1, 1, 1]$, men den optimala lösningen är $[7, 7]$!

Har ni något förslag på en annan lösning?

Exempelproblem 2

- Problemet behöver lösas med en annan strategi*.
- Vi använder **dynamisk programmering**.
- Vi bygger en lista med lösningar för varje $n \leq N$, och använder tidigare element i listan för att bygga nästa element.

*Ibland kan giriga algoritmer ändå användas för att ge en approximativt optimal lösning i polynomisk tid.

Exempelproblem 2

Pseudokod:

minCoins(N, L)
- $L_0 \leftarrow 0$
- **for** $i \leftarrow 1$ **to** N **do**
  - $n_{\text{min}} \leftarrow \infty$
  - **for** $v \in V$ **do**
    - **if** $v > i$ **then**
      - **continue**
    - $n \leftarrow L_{i-v} + 1$
    - **if** $n < n_{\text{min}}$ **then**
      - $n \leftarrow n_{\text{min}}$
  - $L_i \leftarrow n_{\text{min}}$
- **return** L_n

Exempelproblem 2: dynamisk algoritm

$$N = 14,\ V = [10,\ 7,\ 1]$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0														

$$1 = 1$$

$$n_{\min} = 0 + 1 = 1$$

Exempelproblem 2: dynamisk algoritm

$$N = 14,\ \ V = [10,\ 7,\ 1]$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1													

$$2 = 1 + 1$$

$$n_{\text{min}} = 1 + 1 = 2$$

Exempelproblem 2: dynamisk algoritm

$$N = 14,\ \ V = [10,\ 7,\ 1]$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2												

$$3 = 1 + 1 + 1$$

$$n_{\min} = 2 + 1 = 3$$

Exempelproblem 2: dynamisk algoritm

$$N = 14,\ \ V = [10,\ 7,\ 1]$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3											

$$4 = 1 + 1 + 1 + 1$$

$$n_{\min} = 3 + 1 = 4$$

Exemplelproblem 2: dynamisk algoritm

$$N = 14,\ V = [10,\ 7,\ 1]$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4										

$$5 = 1 + 1 + 1 + 1 + 1$$

$$n_{\min} = 4 + 1 = 5$$

Exemplelproblem 2: dynamisk algoritm

$$N = 14,\ \ V = [10,\ 7,\ 1]$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4	5									

$$6 = 1 + 1 + 1 + 1 + 1 + 1$$

$$n_{\min} = 5 + 1 = 6$$

$$N = 14,\ V = [10,\ 7,\ 1]$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4	5	6								

$$7 = 7$$

$$n_{\min} = 0 + 1 = 1$$

Exempelproblem 2: dynamisk algoritm

$$N = 14,\ \ V = [10,\ 7,\ 1]$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4	5	6	1							

$$8 = 7 + 1$$

$$n_{\min} = 1 + 1 = 2$$

$$N = 14,\ V = [10,\ 7,\ 1]$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4	5	6	1	2						

$$9 = 7 + 1 + 1$$

$$n_{\text{min}} = 2 + 1 = 3$$

Exempelproblem 2: dynamisk algoritm

$$N = 14,\ V = [10,\ 7,\ 1]$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4	5	6	1	2	3					

$$10 = 10$$

$$n_{\min} = 0 + 1 = 1$$

Exempelproblem 2: dynamisk algoritm

$$N = 14,\ V = [10,\ 7,\ 1]$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4	5	6	1	2	3	1				

$$11 = 10 + 1$$

$$n_{\text{min}} = 1 + 1 = 2$$

Exempelproblem 2: dynamisk algoritm

$$N = 14,\ V = [10,\ 7,\ 1]$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4	5	6	1	2	3	1	2			

$$12 = 10 + 1 + 1$$

$$n_{\min} = 2 + 1 = 3$$

Exemplelproblem 2: dynamisk algoritm

$$N = 14,\ V = [10,\ 7,\ 1]$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4	5	6	1	2	3	1	2	3		

$$13 = 10 + 1 + 1 + 1$$

$$n_{\text{min}} = 3 + 1 = 4$$

Exempelproblem 2: dynamisk algoritm

$$N = 14,\ \ V = [10,\ 7,\ 1]$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4	5	6	1	2	3	1	2	3	4	

$$14 = 7 + 7$$

$$n_{\mathrm{min}} = 1 + 1 = 2$$

Exempelproblem 2: dynamisk algoritm

$$N = 14,\ \ V = [10,\ 7,\ 1]$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4	5	6	1	2	3	1	2	3	4	2

$$14 = 7 + 7$$

Svar: 2

Notera att vi kan återanvända listan för fler beräkningar.

Exempelproblem 2

- Vi får endast antalet mynt som krävs med denna algoritm.
- Hur kan vi konstruera en lista med valörer?

Exemplelproblem 2

- **Svar**: Vi bygger en till lista med den valör vi väljer i varje steg.
- Då kan vi backtracka för att konstruera listan.

Dynamisk programmering

Dynamisk programmering

- I dynamisk programmering letar man ofta efter någon slags rekursion.
- Man kan sedan bygga en datastruktur som uppfyller denna rekursion.
- I förra lösningen byggde vi en lista L som uppfyllde:

$$L_n = 1 + \min_{v \in V}\{L_{n-v}\}, \; L_0 = 0.$$

Exemplelproblem 3

- Vi har en lista med heltal $L = [1, -4, 15, ...]$ med längd n.
- Vi vill bestämma summan av talen mellan index a och b.
- Vi ska alltså beräkna $\sum_{a \leq i < b} L_i$, (L är nollindexerad).
- Vad blir tidskomplexiteten i värsta fallet?

Exemplelproblem 3

- **Svar**: $O(n)$, värsta fallet: $a = 0, b = n$.
- Går det att förbättra?
- Vi vill nu beräkna k summor, med olika a och b.
- Vad blir tidskomplexiteten i detta fall?

Exemplelproblem 3

- **Svar**: $O(nk)$, vi gör samma sak k gånger.
- Går det att förbättra?

- **Ja!**
- Vi bygger en lista S med delsummor:

$$S_i = \sum_{0 \leq j < i} L_j .$$

Exemplelproblem 3

- Vi kan använda följande rekursion för att bygga S i $O(n)$:

$$S_i = S_{i-1} + L_{i-1}, \;\; S_0 = 0.$$

Exemplelproblem 3

- Vi kan sedan använda S för att beräkna delsummor i konstant tid.

$$\sum_{a \leq i < b} L_i = \sum_{0 \leq i < b} L_i - \sum_{0 \leq i < a} L_i = S_b - S_a.$$

- Vi kan därför beräkna k delsummor i $O(n + k)$.
- Hur många element finns i S?

Läxa: Kattisuppgifter