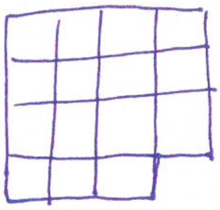
Antag att vi har ett rutnät med $2^n \times 2^n$ rutor, och så tar vi bort rutan i nedre högra hörnet.

(actually, it doesn't matter which square we delete !!)

T ex $n = 2$



Visa att varje sådant rutnät kan täckas helt av L-formade brickor med tre rutor  (för $n \geq 1$).

$\forall n$ $2^n \times 2^n$ can be covered with L-tiles.

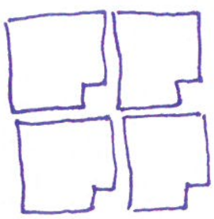
$$\frac{P(0) \quad \forall k\,(P(k) \rightarrow P(k+1))}{\forall n\, P(n)}$$

Bevis genom induktion över n:

Basfall $n = 1$  kan uppenbarligen täckas.

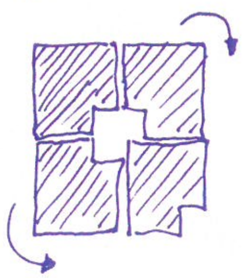
Ind. hypotes: Antag att ett rutnät med sidan 2^k kan täckas. Vi vill nu visa att vi kan täcka ett rutnät med sidan 2^{k+1}.

Ind. steg: Ett rutnät med sidan 2^{k+1} kan konstrueras av 4 st rutnät med sidan 2^k.

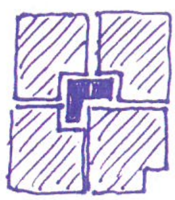


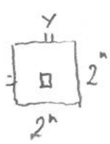
Enl. hypotesen kan alla dessa del-kvadrater täckas.

Rotera nu övre högra och nedre vänstra kvadraten:



Lägg en bricka i det L-formade hål som uppstår, et voilà! Kvadraten är täckt.



Homework: $\forall n \forall x \forall y$  2^n can be covered with L-tiles (rotated)

Defintion av listor av heltal:

<IntList> ::= [] | cons(<Int>, <IntList>)

Funktionen append lägger samman två listor:

append([], x) = x

append(cons(x,y), z) = cons(x, append(y,z))

Funkionen reverse vänder på en lista:

reverse([]) = []

reverse(cons(x,y)) = append(reverse(y), cons(x, [])).

Funktionen length räknar ut längden på en lista:

length([]) = 0

length(cons(x,y)) = 1 + length(y)

Visa att length(x) = length(reverse(x)) för alla listor x.

Vi kommer att använda att length(append(x,y)) = length(x) + length(y)

(bevisades på föreläsningen).

Bevis genom strukturell induktion:

x=[] **Basfall:** length([]) = length(reverse([])) /* def av reverse */

x=cons(a,y) for some a and y. **Ind. steg:** Antag att påståendet gäller för en godtycklig lista y. Betrakta cons(x,y). Vi har:

Ind. hyp.:

length(cons(x,y)) = /* def av length */

= length(y) + 1 = /* induktionshypotes */

= length(reverse(y)) + 1 = /* def av length */

= length(reverse(y)) + length(cons(x, [])) = /* bevisat på föreläsning */

= length(append(reverse(y), cons(x, []))) = /* def av reverse */

= length(reverse(cons(x,y))

Studenterna gör:

1) Definiera binära träd med heltal i löven (t. ex 

) med en grammatik.

Lösning: ⟨BTree⟩ = leaf(⟨Integer⟩) | t(⟨BTree⟩, ⟨BTree⟩)

2) Definiera en rekursiv funktion numleaves som returnerar antalet löv i ett binärt träd.

Lösning:
$$numleaves(leaf(x)) = 1$$
$$numleaves(t(A,B)) = numleaves(A) + numleaves(B)$$

3) Definiera en rekursiv funktion leaves som givet ett binärt träd T returnerar en lista med Ts löv.

Lösning:
$$leaves(leaf(x)) = cons(x, [\,])$$
$$leaves(t(A,B)) = app(leaves(A), leaves(B))$$

4) Bevisa att $numleaves(T) = length(leaves(T))$, för alla binära träd T.

Basfall: $numleaves(leaf(x)) = 1 = 1 + length([\,]) =$
$= length(cons(x, [\,])) = length(leaves(leaf(x)))$

Ind. hypotes: Antag $numleaves(T_1) = length(leaves(T_1))$ o $numleaves(T_2) = length(leaves(T_2))$ för träd T_1 o T_2

Ind. steg: $numleaves(t(T_1,T_2)) = numleaves(T_1) + numleaves(T_2)$
$= length(leaves(T_1)) + length(leaves(T_2)) =$
$= length(app(leaves(T_1), leaves(T_2))) =$