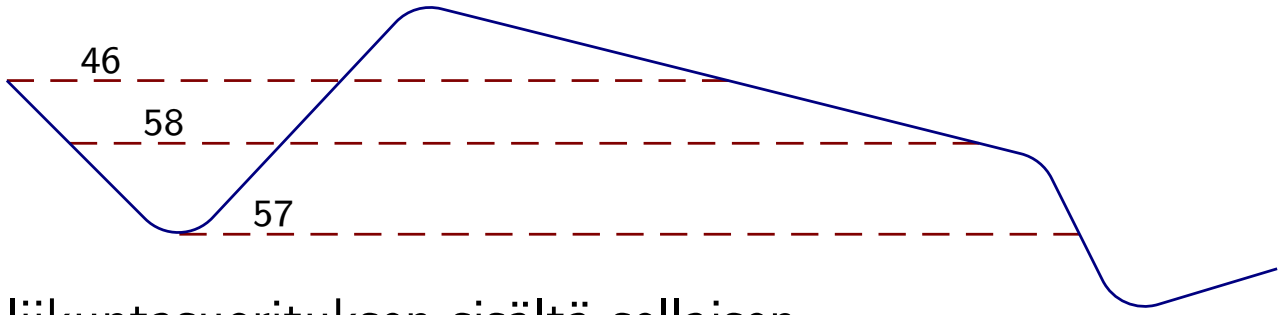
Ohjelmat ”helppo” ja ”nopea” ilman nettoalamäkeä

Epätyhjästä taulukosta `korkeus` on löydettävä mahdollisimman pitkän osuuden, jonka loppukohta on vähintään yhtä korkealla kuin alkukohta, pituus



- esim. pitkän liikuntasuorituksen sisältä sellaisen osuuden pituus, jossa ei ole netto alamäkihyötyä
  - vaikka todellisuus on jatkuva käyrä, korkeustiedot ovat esim. 10m välein
- osuus *kelpaa*, jos ja vain jos sen loppukohta on ainakin yhtä korkealla kuin alkukohta
- vertaamme helppoa mutta hidasta algoritmia keskivaikeaan mutta nopeaan

Helppo algoritmi

- kokeillaan kaikki alkukohdat ja kaikki loppukohdat
- valitaan paras tulos
- yksinkertainen, tehtävää läheisesti muistuttava toimintaperiaate
  $\Rightarrow$ helppo vakuuttautua, että toimii oikein

Toteutus C++:lla

- `korkeus` indeksoidaan `0,...,nn-1`
- `ii` käy läpi kaikki mahdolliset alkukohdat: kaikki kohdat
- `jj` käy läpi kaikki mahdolliset loppukohdat: kaikki kohdat, joille `jj` $\geq$ `ii`
- `paras` pitää kirjaa parhaasta löytyneestä tuloksesta
  - sille on annettava aluksi jokin arvo, joka ei voi vääristää lopputulosta
  - ainakin 0:n pituinen kelpaava osuus on olemassa: pysytään alkukohdassa

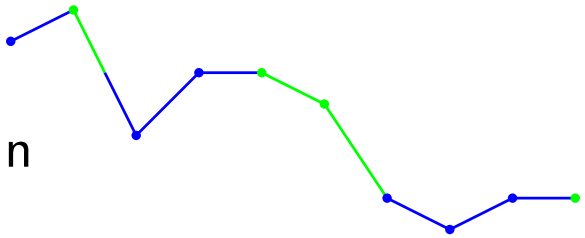
  $\Rightarrow$ voidaan sijoittaa aluksi `paras = 0`
  - `korkeus[ jj ] >= korkeus[ ii ]` testaa, että osuus kelpaa
  - `jj - ii > paras` testaa, että osuus on parempi kuin jo tutkitut

```
unsigned helppo( unsigned korkeus[], unsigned nn ){
  unsigned paras = 0;
  for( unsigned ii = 0; ii < nn; ++ii ){
    for( unsigned jj = ii; jj < nn; ++jj ){
      if( korkeus[ jj ] >= korkeus[ ii ] && jj - ii > paras ){
        paras = jj - ii;
      }
    }
  }
  return paras;
}
```

Ajan kulutus sekunteina

nn	1 000	3 162	10 000	31 623	100 000	316 228	1 000 000
helppo	0	0	0	2	23	238	2 344
nopea	0	0	0	0	0	0	0

Nopean algoritmin peruskäsite: takahuippu



- jos $0 \leq i < n$, niin *i:lle paras loppukohta* on mahdollisimman suuri j, jolle $i \leq j < n$ ja $\texttt{korkeus}[j] \geq \texttt{korkeus}[i]$
  - jos mahdollisimman pitkä osuus on $a \ldots \ell$, niin ℓ on a:lle paras loppukohta
- jos osuus $i \ldots j$ kelpaa ja on olemassa sellainen kohta j', että $j < j' < n$ ja $\texttt{korkeus}[j'] \geq \texttt{korkeus}[j]$, niin osuus $i \ldots j'$ kelpaa ja on pitempi kuin osuus $i \ldots j$

  $\Rightarrow$ parhaalla loppukohdalla i:lle on erityinen ominaisuus:
  sen jälkeen ei enää tule ainakin yhtä korkeita kohtia
- annamme nimen kohdille, joilla on tämä ominaisuus: *takahuippu*

$$\text{takahuippu}(j) \;\Leftrightarrow\; 0 \leq j < n \wedge \forall k\,;\, j < k < n : \texttt{korkeus}[k] < \texttt{korkeus}[j]$$

  $\Rightarrow$ *i:lle paras loppukohta on takahuippu*
- jos $0 \leq j < j' < n$ ja sekä j että j' on takahuippu, niin $\texttt{korkeus}[j] > \texttt{korkeus}[j']$
  - kaikki kohdat j:n jälkeen ovat alempana kuin j, joten myös j' on

  $\Rightarrow$ *seuraava takahuippu on edellistä alempana*

$\Rightarrow$ i:lle paras loppukohta löytyy selaamalla takahuippuja vasemmalta alkaen kunnes ne loppuvat tai niiden korkeus alittaa i:n korkeuden

- jos $0 \leq i < i' < n$, niin joko $\texttt{korkeus}[i'] \geq \texttt{korkeus}[i]$ tai $\texttt{korkeus}[i'] < \texttt{korkeus}[i]$
- jos $\texttt{korkeus}[i'] \geq \texttt{korkeus}[i]$, niin
  - jos $i' \ldots j$ on kelpaava osuus, niin $\texttt{korkeus}[i] \leq \texttt{korkeus}[i'] \leq \texttt{korkeus}[j]$

  $\Rightarrow$ $i \ldots j$ on kelpaava osuus

  $\Rightarrow$ i' ei voi olla mahdollisimman pitkän kelpaavan osuuden alkukohta
  - ei tarvitse etsiä i':lle parasta loppukohtaa
- jos $\texttt{korkeus}[i'] < \texttt{korkeus}[i]$, niin i':lle paras loppukohta on sama tai alempana kuin i:lle paras loppukohta
  - jokainen paras loppukohta on takahuippu
  - takahuiput ovat laskevassa järjestyksessä

  $\Rightarrow$ i':lle paras loppukohta on sama tai enemmän oikealla kuin i:lle paras loppukohta
  - jos se on sama, niin sitä ei tarvitse tutkia i':lle, koska i on sille parempi alkukohta

  $\Rightarrow$ voidaan jatkaa takahuippujen selaamista oikealle siitä mihin jäätiin
- viimeisen kohdan jälkeen ei tule mitään kohtia

  $\Rightarrow$ ei tule ainakin yhtä korkeita kohtia

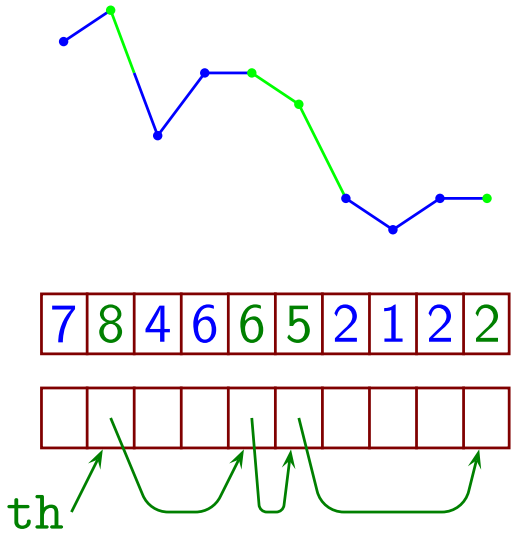
  $\Rightarrow$ *viimeisessä kohdassa on takahuippu*

Nopea algoritmi

- etsitään takahuiput ja talletetaan ne listaksi, jota voi selata vasemmalta oikealle
- selataan alkukohdat ja ym. lista tahdistetusti

Nopea algoritmi C++:lla

```
unsigned nopea( unsigned korkeus[], unsigned nn ){
```



- etsitään takahuiput selaamalla `korkeus` takaperin
  - `th` on korkein löydetty kohta
  - ensimmäiseksi löydetään viimeinen kohta, joten alustetaan `th = nn-1`
  - kohta on takahuippu, jos ja vain jos se on viimeinen kohta tai korkeampi kuin korkein aiemmin löydetty kohta
  - `seuraava_th[ ii ]` on takahuippua `ii` seuraava (eli uusin sitä ennen löydetty) takahuippu

```
  unsigned th = nn-1, *seuraava_th = new unsigned[ nn ];
  for( unsigned ii = nn; ii--; ){
    if( korkeus[ ii ] > korkeus[ th ] ){
      seuraava_th[ ii ] = th; th = ii;
    }
  }
```

- etsitään paras tulos selaamalla aloituskohdat ja takahuiput etuperin
  - `for`-silmukka selaa aloituskohdat
  - `while`-silmukka selaa takahuippuja kunnes ne loppuvat (`th == nn-1`) tai ohitetaan `ii`:lle paras loppukohta (`while`:n ehto ei toteudu)
  - pidetään kirjaa parhaasta selatun osuuden pituudesta

```
unsigned paras = 0;
for( unsigned ii = 0; ii < nn; ++ii ){
  while( korkeus[ th ] >= korkeus[ ii ] ){
    if( th - ii > paras ){ paras = th - ii; }
    if( th == nn-1 ){ delete[] seuraava_th; return paras; }
    th = seuraava_th[ th ];
  }
}
```

- on helppo nähdä, että algoritmi kokeilee vain korkeusehdon toteuttavia osuuksia ja valitsee kokeilemistaan parhaan
- lopussa on rivi, jolle tullaan vain jos $n = 0$

```
delete[] seuraava_th; return paras; }
```

- voiko (jokainen) kaikkein paras osuus $a \ldots \ell$ jäädä kokeilematta?
  - `ii` kokeilee jokaisen alkukohdan

  $\Rightarrow$ `ii` kokeilee parhaan osuuden alkukohdan a
  - parhaan osuuden loppukohta ℓ on takahuippu
  - jos $\texttt{ii} < a$, niin $\texttt{korkeus}[\ell] < \texttt{korkeus}[\texttt{ii}]$ (miksi?)

  $\Rightarrow$ `th` ei ohita kohtaa ℓ kun $\texttt{ii} < a$

  $\Rightarrow$ suorituksen aikana on hetki, jolloin $\texttt{ii} = a$ ja $\texttt{th} = \ell$

  $\Rightarrow$ ohjelma tutkii jokaisen kaikkein parhaan osuuden