Olgu antud (reaalarvu)maatriksid $A_1, \ldots, A_n$.

Olgu maatriksi A_i dimensioonid $d_{i-1} \times d_i$.

Mitu arvude korrutamist tuleb teha korrutise $A_1 \cdots A_n$ leidmisel?

$m \times n$ ja $n \times p$ maatriksite korrutamisel tekib $m \times p$ maatriks, arvude korrutamisi tuleb mnp.

Kolme maatriksi korral (A_1, A_2, A_3) võib nende korrutist leida kahel viisil: $(A_1A_2)A_3$ ja $A_1(A_2A_3)$. Mõlemad annavad sama tulemuse (maatriksite korrutamine on assotsiatiivne).

Esimesel juhul on arvude korrutamiste arv $d_0d_1d_2 + d_0d_2d_3$.

Teisel juhul on arvude korrutamiste arv $d_1d_2d_3 + d_0d_1d_3$.

Need on üldiselt erinevad.

Milline on korrutamiste minimaalne arv? Kuidas seda leida?

Mitmel erineval viisil on võimalik avaldises $A_1 \cdots A_n$ sulud paigutada?

Vaatame viimast korrutustehet: $A_1 \cdots A_n = (A_1 \cdots A_k) \cdot (A_{k+1} \cdots A_n)$.

Olgu T_i sulupaigutusviiside arv i elemendi korrutamisel. Siis n elemendi korrutamiseks, kui viimasena korrutatakse $A_1 \cdots A_k$ ja $A_{k+1} \cdots A_n$, on $T_k T_{n-k}$ võimalust.

Seega $T_n = \sum_{k=1}^{n-1} T_k T_{n-k}$. Peale selle $T_1 = 1$.

T_n on eksponentsiaalne n suhtes (induktsiooniga on lihtne näidata, et $T_n \geqslant 2^n T_1$). Kõigi sulupaigutusvõimaluste läbivaatus oleks väga kulukas.

Uurime optimaalse korrutamisstrateegia struktuuri. Olgu k selline indeks, et viimasena korrutatakse $A_1 \cdots A_k$ ja $A_{k+1} \cdots A_n$.

Sel juhul annab maatriksite $A_1, \ldots, A_n$ optimaalne korrutamisstrateegia meile ka optimaalse korrutamisstrateegia maatriksite $A_1, \ldots, A_k$ ja $A_{k+1}, \ldots, A_n$ jaoks.

Tõepoolest, kui leiduks parem strateegia maatriksite $A_1, \ldots, A_k$ korrutamiseks, siis võiksime $A_1 \cdots A_n$ leidmisel korrutise $A_1 \cdots A_k$ hoopis selle strateegiaga leida (ja muidu kasutada ikka $A_1 \cdots A_n$ leidmise optimaalset strateegiat). Korrutamiste arv väheneks sellest.

Olgu $K_{i..j}$ minimaalne vajalik korrutamiste arv $A_i \cdots A_j$ leidmiseks (fikseeritud $d_0, d_1, \ldots, d_n$ jaoks).

$A_1 \cdots A_n$ leidmiseks minimaalse vajaliku korrutamiste arvu $K_{1..n}$ leidmiseks:

- Leiame $K_{1..1}, K_{1..2}, \ldots K_{1..n-1}$ ja $K_{n..n}, K_{n-1..n}, \ldots K_{2..n}$.
- Leiame, mitu korrutustehet tuleb teha $A_1 \cdots A_k$ ja $A_{k+1} \cdots A_n$ korrutamiseks (iga k jaoks). Vastus: $d_0 d_k d_n$.
- Vajalik korrutamiste arv on

$$\min_{1 \leqslant k \leqslant n-1} \left(K_{1..k} + K_{k+1..n} + d_0 d_k d_n \right) \ .$$

$d_0, d_1, \ldots, d_n$ — globaalsed muutujad.

leiaK_rek(i, j) on

```
1  if i = j then return 0
2  tul := ∞
3  for k := i to j − 1 do
4      tk := leiaK_rek(i, k) + leiaK_rek(k + 1, j) + d_{i−1} d_k d_j
5      if tk < tul then tul := tk
6  return tul
```

Keerukus:

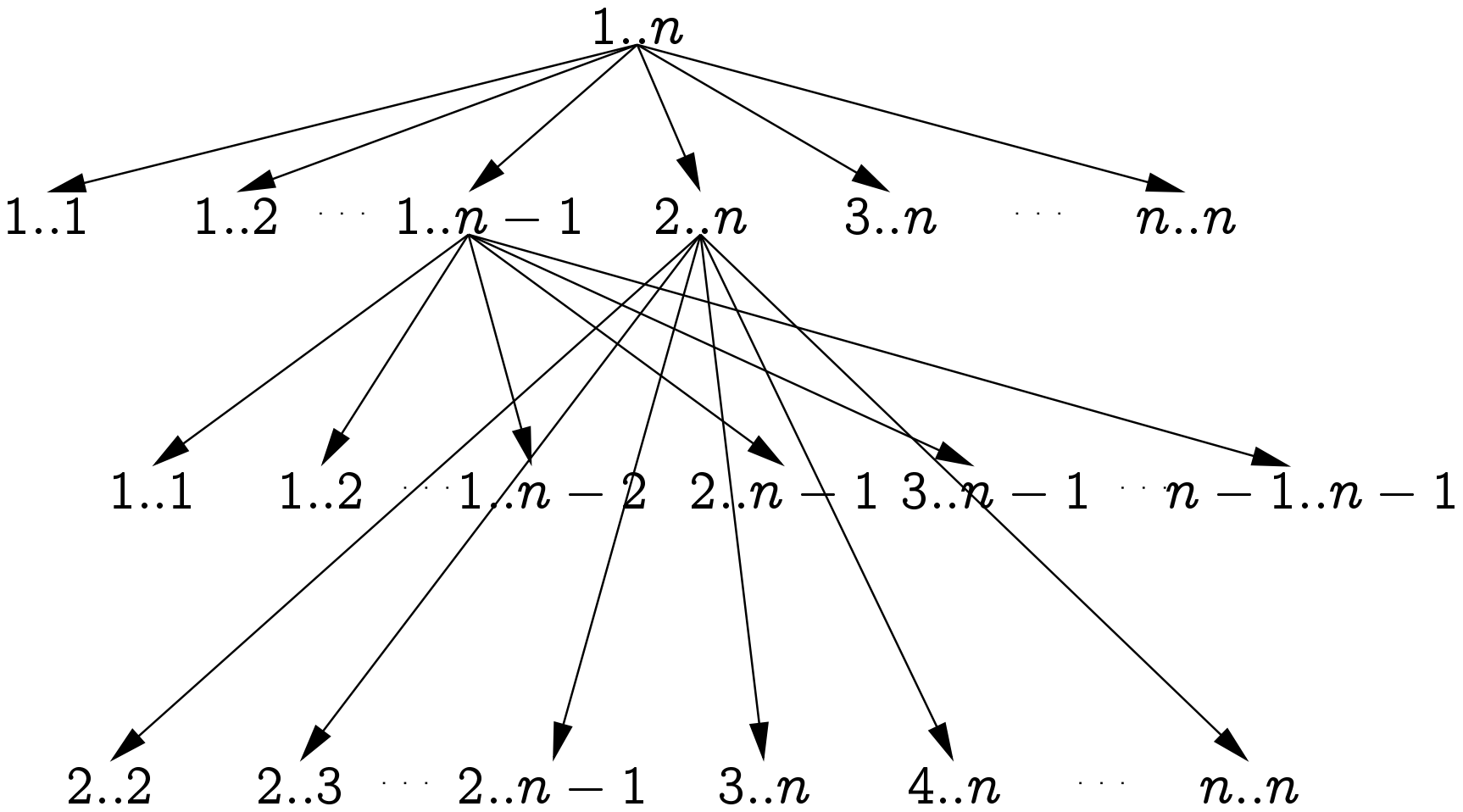
$$T(1) = \Theta(1)$$

$$T(n) = \sum_{k=1}^{n-1} T(k) + T(n-k) + \Theta(n)$$

Eksponentsiaalne (induktsiooniga on lihtne näidata, et $T(n) \geqslant 2^{n-1}T(1)$).

Vaatame rekursioonipuud:

Fragment puust:



Väga palju korratakse.

leiaK_rek argumentideks on i ja j, kus $1 \leqslant i \leqslant n$ ja $i \leqslant j \leqslant n$. Kokku on $\Theta(n^2)$ erinevat võimalikku sisendit leiaK_rek-le.

Idee: kogume leiaK väärtused tabelisse. Kui on tarvis leida leiaK mingil kohal, kus on juba leitud, siis ei arvuta uuesti, vaid võtame tabelist.

initsialiseeriK_memo(n) on

```
1   for i := 1 to n do
2       for j := i to n do
3           K[i, j] := −1
                    --- −1 tähistab „initsialiseerimata“
```

leiaK_memo(i, j) on

```
1  if K[i,j] ≠ −1 then return K[i,j]
2  if i = j then K[i,j] := 0; return 0
3  tul := ∞
4  for k := i to j − 1 do
5      tk := leiaK_memo(i,k)+
                    leiaK_memo(k+1,j) + d_{i−1} d_k d_j
6      if tk < tul then tul := tk
7  K[i,j] := tul
8  return tul
```

Kõigepealt kutsuda initsialiseeriK_memo(n), seejärel leiaK_memo$(1, n)$.

Keerukus: iga paari (i, j) jaoks täidetakse funktsiooni põhiosa 1 kord. Põhiosa on keerukusega $\Theta(j - i)$. Paare on $\Theta(n^2)$. Kokku on

$$\sum_{i=1}^{n} \sum_{j=i}^{n} \sum_{k=i}^{j-1} \Theta(1) = \Theta(n^3) \ .$$

Meil oli

$$K_{i..j} = \min_{i \leqslant k \leqslant j-1} \big(K_{i..k} + K_{k+1..j} + d_{i-1} d_k d_j \big),$$

s.t. $K_{i..j}$ leidmiseks on vaja teada ainult selliseid $K_{k..l}$-e, kus $l - k < j - i$.

Me võime tabeli $K[i, j]$ täita $j - i$ kasvamise järjekorras.

leiaK_tabel(n) on

```
1  for i = 1 to n do K[i, i] := 0
2  for vahe := 1 to n − 1 do
3      for i := 1 to n − vahe do
4          j := i + vahe
5          K[i, j] := ∞
6          for k := i to j − 1 do
7              tk := K[i, k] + K[k + 1, j] + d_{i−1} d_k d_j
8              if tk < K[i, j] then K[i, j] := tk
9  return K[1, n]
```

Oleme leidnud, kui mitu korrutamistehet vaja läheb.

Aga kuidas tegelikult korrutada? Meil oli

$$K_{i..j} = \min_{i \leqslant k \leqslant j-1} \big(K_{i..k} + K_{k+1..j} + d_{i-1} d_k d_j \big),$$

seega viimasena tuleb korrutada $A_1 \cdots A_k$ ja $A_{k+1} \cdots A_n$, kus k on selline, mis realiseerib selle miinimumi.

Piisab, kui iga $K[i,j]$ jaoks salvestame ka k, mis miinimumi realiseeris.

leia_korrutusviis(n) on

```
 1  for i = 1 to n do K[i,i] := 0
 2  for vahe := 1 to n − 1 do
 3      for i := 1 to n − vahe do
 4          j := i + vahe
 5          K[i,j] := ∞
 6          for k := i to j − 1 do
 7              tk := K[i,k] + K[k+1,j] + d_{i−1} d_k d_j
 8              if tk < K[i,j] then
 9                  K[i,j] := tk
10                  M[i,j] := k
11  return M
```

Olgu $A_1, \ldots, A_n$ ja $d_0, \ldots, d_n$ globaalsed muutujad. Siis korrutise $A_1, \ldots, A_n$ leidmiseks:

korrutaM$(1, n,$ leia_korrutusviis$(n))$.

korrutaM(i, j, M) on

```
1  if i = j then return A_i
2  B := korrutaM(i, M[i, j], M)
3  C := korrutaM(M[i, j] + 1, j, M)
4  return B · C          --- maatriksite korrutamine
```

Näide: olgu meil 6 maatriksit mõõtmetega $2, 6, 4, 5, 2, 3, 4$.

Siis K ja M on

$i\backslash j$	1	2	3	4	5	6
1	0	48	88	104	116	140
2		0	120	88	124	160
3			0	40	64	96
4				0	30	64
5					0	24
6						0

$i\backslash j$	2	3	4	5	6
1	1	2	2	4	5
2		2	2	4	4
3			3	4	4
4				4	4
5					5

Kiireim korrutamisviis: $(((A_1A_2)(A_3A_4))A_5)A_6$.

Eeltoodud lahendusviisi nimetatakse *dünaamiliseks programmeerimiseks*. Sõna „programmeerimine“ viitab tabelite kasutamisele optimaalse lahenduse leidmisel.

Dünaamilise programmeerimise kasutamiseks peavad ülesandel olema järgmised kaks omadust:

- Optimaalne alamstruktuur.
- Kattuvad alamülesanded.

Optimaalne alamstruktuur — ülesande optimaalne lahendus pannakse mingil viisil kokku tema alamülesannete optimaalsetest lahendustest.

Seejuures tuleb kuidagi fikseerida, millised on võimalikud alamülesanded.

- Meil oli: alamülesande määrasid i ja j, kus $1 \leqslant i \leqslant j \leqslant n$.

Kattuvad alamülesanded — erinevate alamülesannete alamülesanded peavad suures osas samad olema.

Kõigi võimalike alamülesannete hulk ei tohi olla suur.

- Meil oli selle hulga võimsuseks $\Theta(n^2)$.

See lubab meil kõigi alamülesannete vastused salvestada peale nende esmakordset leidmist.

Salvestamiseks nägime eespool kahte võimalust.

Olgu s sõne pikkusega m. Olgu $1 \leqslant i_1 < i_2 < \cdots < i_k \leqslant m$.

Sõne $s[i_1]s[i_2]\cdots s[i_k]$ nimetame s-i *osasõneks*.

Näiteks „abdf“ on sõne „abcdefgh“ osasõne.

Seda, et u on s-i osasõne, tähistame $u \leqslant s$.

Ülesanne: antud sõned s ja t. Leida nende pikim ühine osasõne.

(Rakendus: `/usr/bin/diff`)

Lihtne on kontrollida, kas üks sõne on teise sõne osasõne.

Järgnev funktsioon kontrollib, kas u on s-i osasõne.

```
1  n := |s|; m := |u|; j := 1
2  for i := 1 to n do
3      if s[i] = u[j] then j := j + 1
4      if j = m + 1 then break
5  return j = m + 1
```

See funktsioon leiab u „vasakpoolseima“ esinemise s-s.

t kõigi $2^{|t|}$ osasõne jaoks kontrollimine, kas tegemist on ka s-i osasõnega, võtaks väga kaua aega.

Uurime osasõneks olemise struktuuri lähemalt.

Tähistagu $P\ddot{U}(s,t)$ sõnede s ja t pikimat ühist osasõnet.

Siis $P\ddot{U}(\epsilon,t) = P\ddot{U}(s,\epsilon) = \epsilon$.

Kui $s_1 \sqsubset s$ ja $t_1 \sqsubset t$, siis $|P\ddot{U}(s_1,t_1)| \leqslant |P\ddot{U}(s,t)|$.

Kui $s \neq \epsilon$, siis olgu $s' := s[1 \ldots |s| - 1]$.

Olgu $u = P\ddot{U}(s,t)$.

Lause. Kui $s[|s|] = t[|t|]$ (olgu selleks ühiseks täheks x), siis $u[|u|] = x$.

Tõestus. Oletame vastuväiteliselt, et $u[|u|] \neq x$. Kuna $u \leqslant s$, siis ka $u \leqslant s'$, sest u viimane täht ei saa vastata s-i viimasele tähele. Samuti $u \leqslant t'$. Järelikult $ux \leqslant s$ ja $ux \leqslant t$, seega pole u pikim s-i ja t ühine osasõne.

Seega, kui $s[|s|] = t[|t|] = x$, siis $P\ddot{U}(s,t) = P\ddot{U}(s',t')x$.

Olgu s, t, u sellised sõned, et $s[|s|] \neq t[|t|]$. Peale selle olgu

- $s[|s|] \neq u[|u|]$;
- $u \leqslant s$ ja $u \leqslant t$.

Siis $u \leqslant s'$ ja $u \leqslant t$.

Seega, kui $P\ddot{U}(s,t)$ viimane täht pole $s[|s|]$, siis $P\ddot{U}(s,t) = P\ddot{U}(s',t)$.

Analoogiliselt, kui $P\ddot{U}(s,t)$ viimane täht pole $t[|t|]$, siis $P\ddot{U}(s,t) = P\ddot{U}(s,t')$.

Vähemalt üks neist kahest variandist peab esinema.

$$P\ddot{U}(s,t) = \begin{cases} \epsilon, & \text{kui } s = \epsilon \text{ või } t = \epsilon \\ P\ddot{U}(s',t')s[|s|], & \text{kui } s \neq \epsilon, t \neq \epsilon, s[|s|] = t[|t|] \\ P\ddot{U}(s',t), & \text{kui } s \neq \epsilon, t \neq \epsilon, s[|s|] \neq t[|t|], \\ & \quad |P\ddot{U}(s',t)| \geqslant |P\ddot{U}(s,t')| \\ P\ddot{U}(s,t'), & \text{kui } s \neq \epsilon, t \neq \epsilon, s[|s|] \neq t[|t|], \\ & \quad |P\ddot{U}(s',t)| < |P\ddot{U}(s,t')| \end{cases}$$

$|P\ddot{U}(s,t)|$ leidmiseks leiame $|P\ddot{U}(s[1\ldots i],t[1\ldots j])|$ kõigi $i \in \{0,\ldots,|s|\}$ ja $j \in \{0,\ldots,|t|\}$ jaoks. Keerukus: $\Theta(mn)$, kus $m = |s|$ ja $n = |t|$.

Kui P on massiiv, kus $P[i,j] = |P\ddot{U}(s[1\ldots i], t[1\ldots j])|$, siis leia_PÜ$(m,n)$ leiab $P\ddot{U}(s,t)$. leia_PÜ(i,j) on

```
1  if i = 0 or j = 0 then return ϵ
2  if s[i] = t[j] then
3      return leia_PÜ(i − 1, j − 1)s[i]
4  if P[i − 1, j] ⩾ P[i, j − 1] then
5      return leia_PÜ(i − 1, j)
6  else
7      return leia_PÜ(i, j − 1)
```

S.t. vaatame, millist varianti (eelmiselt kilelt) kasutati $P\ddot{U}(s[1\ldots i], t[1\ldots j])$ leidmiseks.

Massiivi P täitmiseks:

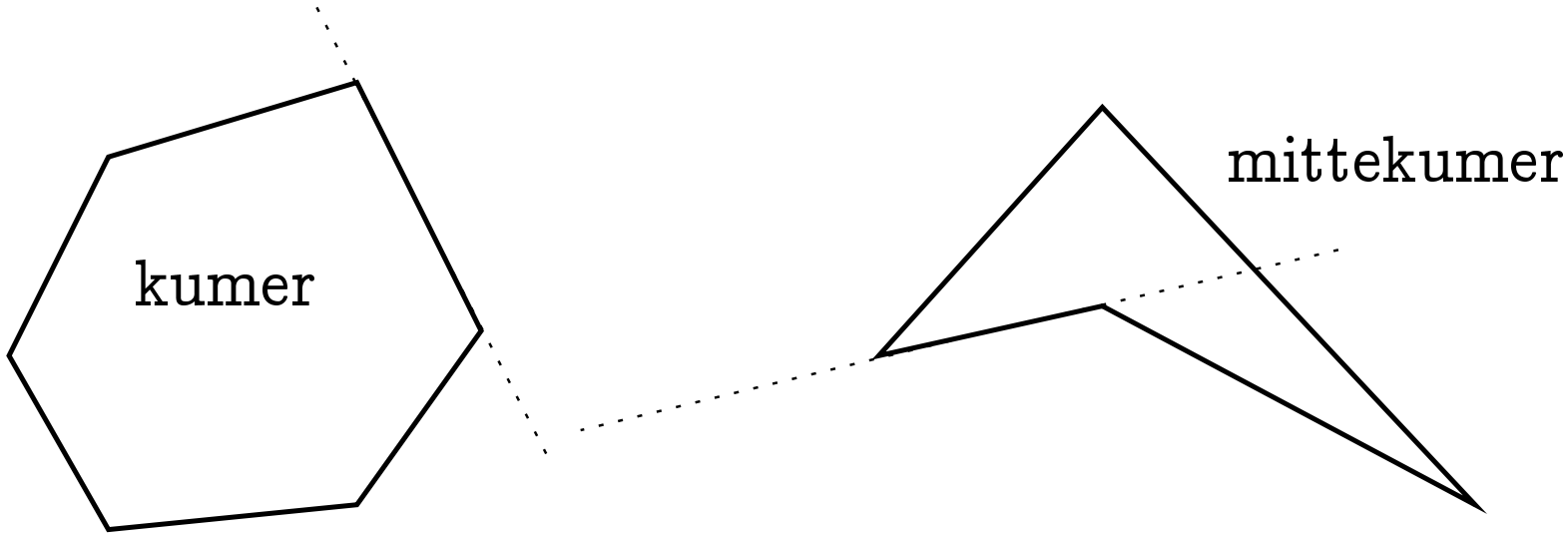
```
1   m := |s|; n := |t|
2   for i := 0 to m do
3       for j := 0 to n do
4           if i = 0 or j = 0 then
5               P[i, j] := 0
6           else if s[i] = t[j] then
7               P[i, j] := P[i − 1, j − 1] + 1
8           else
9               P[i, j] := max(P[i − 1, j], P[i, j − 1])
```

Näide: Olgu $s =$ „abcbdab“ ja $t =$ „bdcaba“. Siis P on

$i \backslash j$	0	1 b	2 d	3 c	4 a	5 b	6 a
0	0	0	0	0	0	0	0
1 a	0	0	0	0	1	1	1
2 b	0	1	1	1	1	2	2
3 c	0	1	1	2	2	2	2
4 b	0	1	1	2	2	3	3
5 d	0	1	2	2	2	3	3
6 a	0	1	2	2	3	3	4
7 b	0	1	2	2	3	4	4

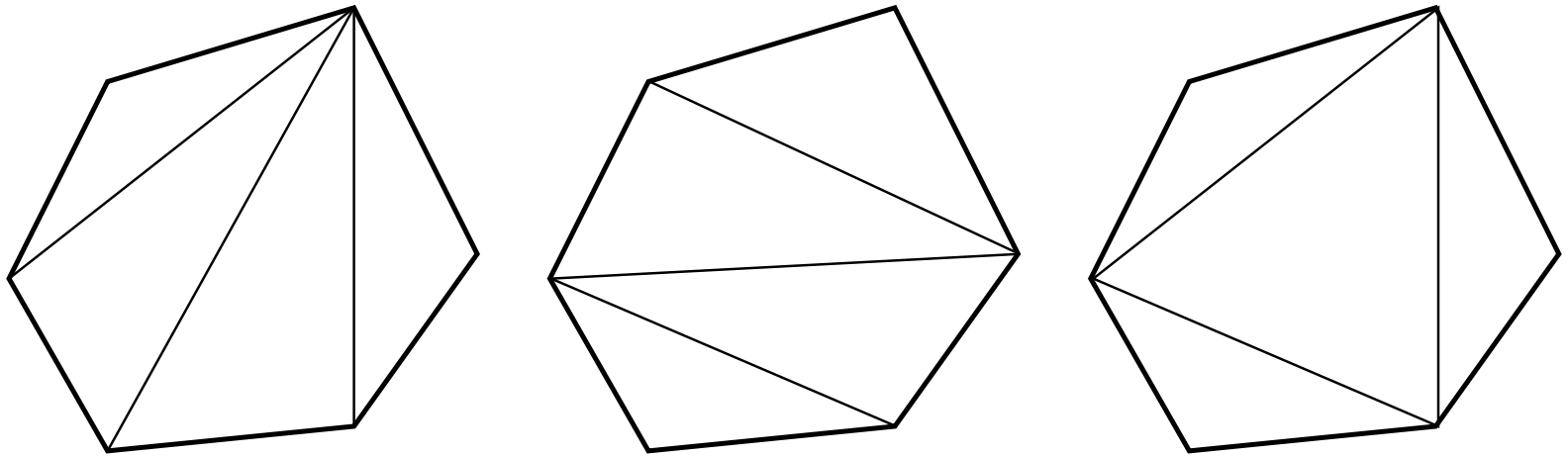
$P\ddot{U}(s,t) =$ „bcba“
või
$P\ddot{U}(s,t) =$ „bcab“
või
$P\ddot{U}(s,t) =$ „bdab“

Olgu tasandil antud kumer hulknurk $A_1 A_2 \cdots A_n$. Olgu A_i koordinaadid (x_i, y_i).



Hulknurk on kumer, kui ta jääb iga oma küljega määratud sirgest ühele poole.

Hulknurga *triangulatsioon* on tema tükeldus sellisteks kolmnurkadeks, mille tippudeks on selle hulknurga tipud.



n-nurga triangulatsioon koosneb $(n-2)$-st kolmnurgast.

Olgu iga kolmnurga, mille tippudeks on mingid kolm tippudest $A_1, \ldots, A_n$, jaoks defineeritud tema *kaal*.

S.t. olgu antud funktsioon W, mis võtab kolm argumenti vahemikust 1 kuni n ja tagastab mingi reaalarvu.

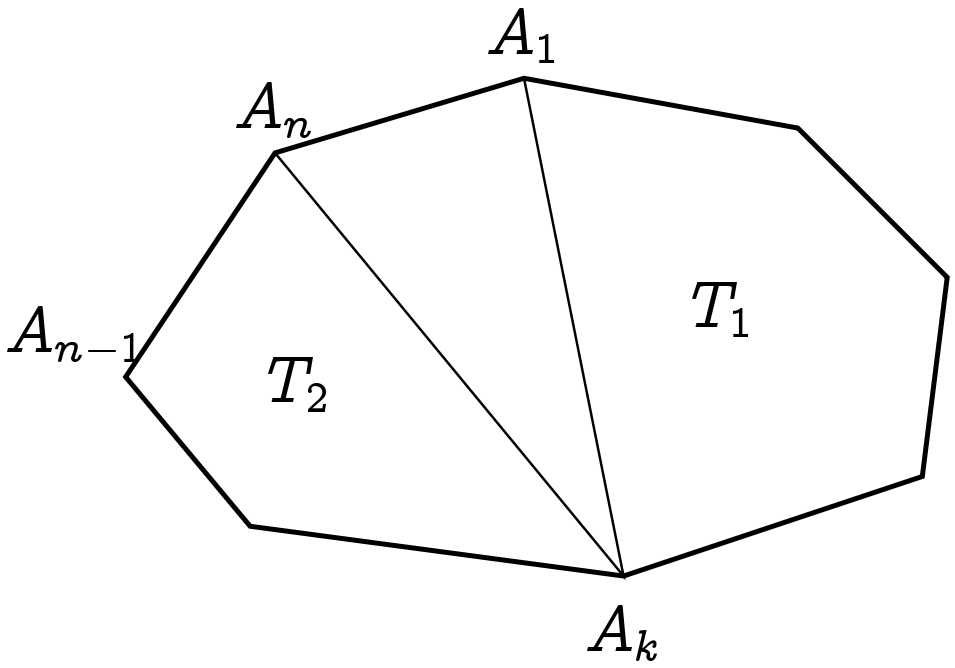
Olgu triangulatsiooni kaal temasse kuuluvate kolmnurkade kaalude summa.

Ülesanne: leida minimaalse kaaluga triangulatsioon.

Vaatame mingit minimaalse kaaluga triangulatsiooni T. Tema kaalu tähistame $w(T)$.

Vaatame hulknurga külge A_nA_1. See külg on triangulatsioonis mingi kolmnurga küljeks.

Olgu selle kolmnurga kolmas tipp A_k.



Siis defineerib T ka hulknurga $A_1A_2\cdots A_k$ mingi triangulatsiooni T_1 ja hulknurga $A_kA_{k+1}\cdots A_n$ mingi triangulatsiooni T_2.

Seejuures $w(T) = W(1,k,n) + w(T_1) + w(T_2)$.

Seega on T_1 ja T_2 minimaalse kaaluga.

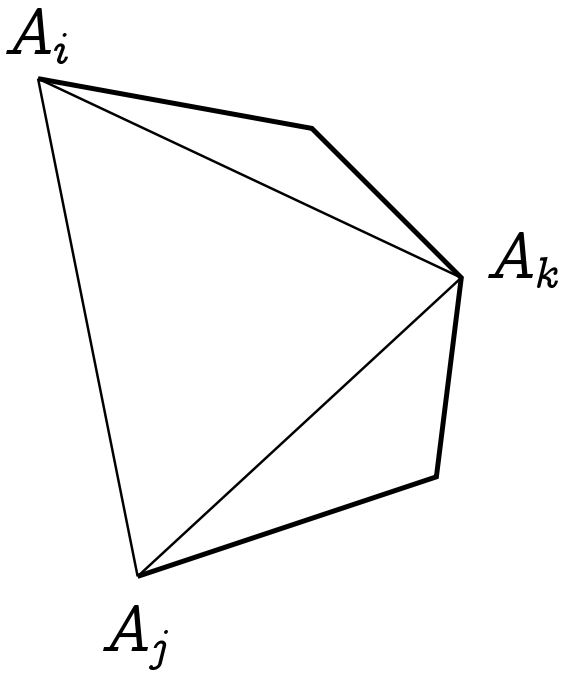
Leidmaks $A_1A_2\cdots A_n$ minimaalse kaaluga triangulatsiooni, tuleb meil leida hulknurkade $A_1A_2\cdots A_k$ ja $A_kA_{k+1}\cdots A_n$ triangulatsioonid. Seejuures $k \in \{3,\ldots,n-2\}$ ning

$$w(A_1A_2\cdots A_n) = \min\big(W(1,2,n) + w(A_2A_3\cdots A_n),$$
$$\min\{W(1,k,n)+w(A_1A_2\cdots A_k)+w(A_kA_{k+1}\cdots A_n) \,:\, 3 \leqslant k \leqslant n-2\},$$
$$W(1,n-1,n) + w(A_1A_2\cdots A_{n-1})\big),$$

kus w, rakendatuna hulknurgale, tähistab tema min. kaaluga triangulatsiooni kaalu.

Leidmaks triangulatsiooni ennast, tuleb meeles pidada, millise koha pealt miinimum saadi.

Üldjuhul, $A_iA_{i+1}\cdots A_j$ min. kaaluga triangulatsiooni kaalu leidmine käib järgmiselt (siin $1 \leqslant i < j \leqslant n$ ja $j - i \geqslant 2$):



$$w(A_iA_{i+1}\cdots A_j) = \min\big(W(i,i{+}1,j){+}w(A_{i+1}A_{i+2}\cdots A_j),$$
$$\min\{W(i,k,j){+}w(A_iA_{i+1}\cdots A_k){+}w(A_kA_{k+1}\cdots A_j)\ :\ i{+}2 \leqslant k \leqslant j{-}2\},$$
$$W(i,j-1,j) + w(A_iA_{i+1}\cdots A_{j-1})\big)$$

Leidmaks $w(A_1 A_2 \cdots A_n)$-i, tuleb järelikult leida suurused $w[i,j]$, kus $1 \leqslant i < j \leqslant n$, $j - i \geqslant 2$ ja $w[i,j]$ tähistab suurust $w(A_i A_{i+1} \cdots A_j)$.

Leidmine käib sarnaselt optimaalse maatriksitekorrutusviisi leidmisele.