

# Algorithmen 1

## Übung 2 Aufgabe B01.A1, amortisierte Analyse



# Organisatorisches

- Blatt 2, Aufgabe 1, erster Algo
  - schlechte Aufgabenstellung, zu schwierig
  - $\Rightarrow$  volle Punktzahl + Bonuspunkte (siehe Vorlesung 4)
  - Gleich: Vorstellung der Lösung
- Zusätzliche Tutorien zum Nachholen / Klausurvorbereitung im August
  - mehr Infos demnächst
- Update der Pseudocode Richtlinien

# Übung 2: Aufgabe 1: ALGONE

Gegeben:

```
algOne( $x: \mathbb{N}, y: \mathbb{N}$ ) // Annahme: initial  $x \leq y$   
  if  $x = 0 \vee x = y$  then  
    return 1  
  return algOne( $x-1, y-1$ ) + algOne( $x, y-1$ )
```

Gesucht:

- „Problemgröße“  $n$
- Rekurrenz in Abh. von  $n$
- Laufzeit in Abh. von  $n$

# Übung 2: Aufgabe 1: ALGONE

Gegeben:

```

algOne(x: ℕ, y: ℕ) // Annahme: initial x ≤ y
| if x = 0 ∨ x = y then
| | return 1
| return algOne(x-1, y-1)
  
```

Gesucht:

- „Problemgröße“  $n$
- Rekurrenz in Abh. von  $n$
- Laufzeit in Abh. von  $n$

```

algTwo(x, y) // Annahme: initial x ≤ y
| if x ≥ y then
| | return x
| total := algTwo(x, ⌊ $\frac{y+3x}{4}$ ⌋)
| total += algTwo(⌊ $\frac{y+3x}{4}$ ⌋, ⌊ $\frac{y+x}{2}$ ⌋)
| total += algTwo(⌊ $\frac{y+x}{2}$ ⌋, ⌊ $\frac{3y+x}{4}$ ⌋)
| return total
  
```

- „Problemgröße“  $n := \max\{0, y - x\}$ 
  - $n' = 1/4 \cdot n$
- $T(n) = 3T(n/4) + c_1, T(0) = c_2$
- $T(n) \in \Theta(n^{\log_4(3)})$

# Übung 2: Aufgabe 1: ALGONE

Gegeben:

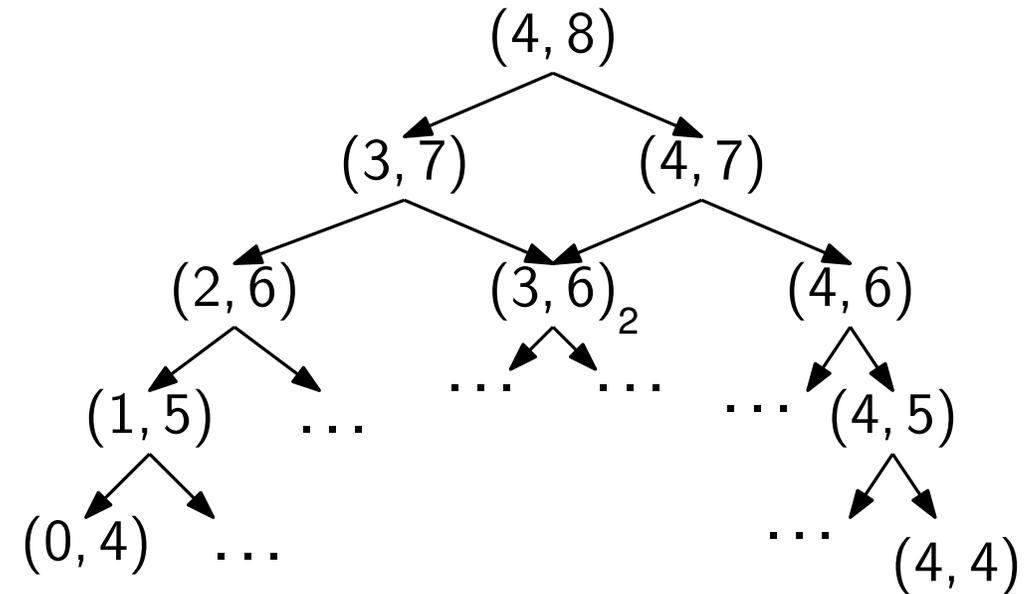
```

algOne( $x: \mathbb{N}, y: \mathbb{N}$ ) // Annahme: initial  $x \leq y$ 
  if  $x = 0 \vee x = y$  then
    return 1
  return algOne( $x-1, y-1$ ) + algOne( $x, y-1$ )
  
```

Gesucht:

- „Problemgröße“  $n$
- Rekurrenz in Abh. von  $n$
- Laufzeit in Abh. von  $n$

**Und bei `algOne`?**



Das schaut kompliziert aus :(

## Plan

- (grobe) obere Schranke in  $y$
- im Anschluss: genaue Analyse

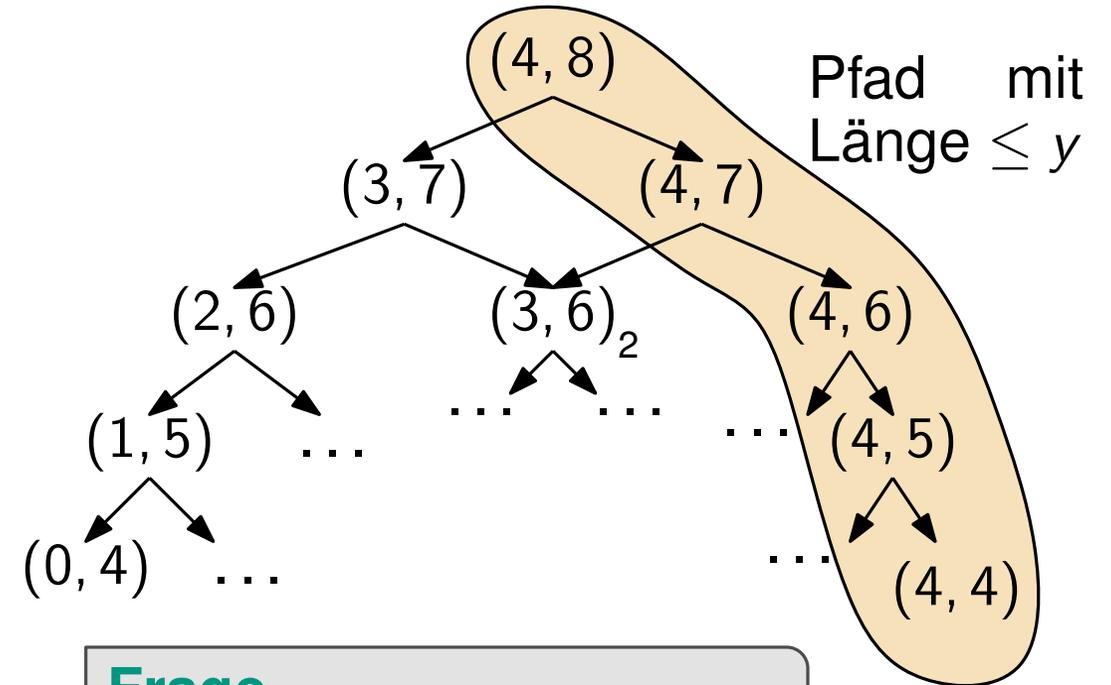
# Ungenauere obere Schranke ALGONE

```

algOne( $x: \mathbb{N}, y: \mathbb{N}$ ) // Annahme: initial  $x \leq y$ 
  if  $x = 0 \vee x = y$  then
    return 1
  return algOne( $x-1, y-1$ ) + algOne( $x, y-1$ )
  
```

## Beobachtung

- $y$  sinkt in jedem Aufruf um 1
- Länge der Pfade im Rekursionsbaum maximal  $y$
- Pro Aufruf: zwei Rekursive Aufrufe
- #Knoten in Rekursionsbaum  $\leq 2 \cdot 2^y$
- Für „Problemgröße“  $n := y$ : Laufzeit in  $O(2^n)$



## Frage

Ist diese Analyse gut?

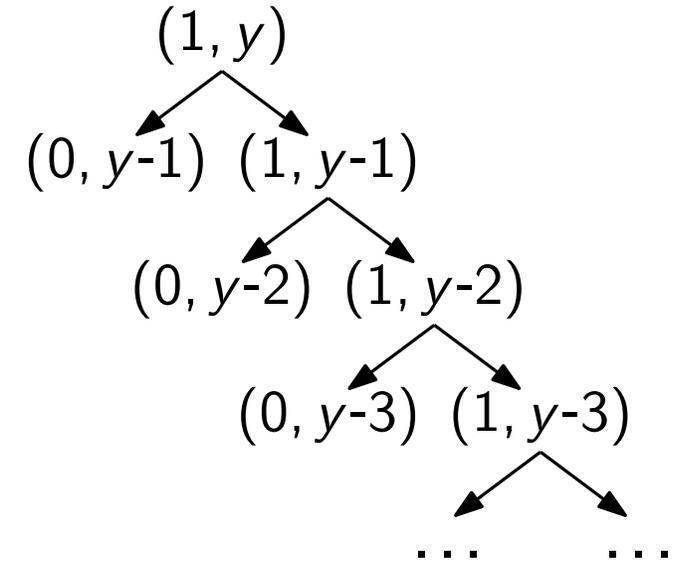
# Ungenauere obere Schranke ALGONE

```

algOne( $x: \mathbb{N}, y: \mathbb{N}$ ) // Annahme: initial  $x \leq y$ 
  if  $x = 0 \vee x = y$  then
    return 1
  return algOne( $x-1, y-1$ ) + algOne( $x, y-1$ )
  
```

## Beobachtung

- $y$  sinkt in jedem Aufruf um 1
- Länge der Pfade im Rekursionsbaum maximal  $y$
- Pro Aufruf: zwei Rekursive Aufrufe
- #Knoten in Rekursionsbaum  $\leq 2 \cdot 2^y$
- Für „Problemgröße“  $n := y$ : Laufzeit in  $O(2^n)$



### Frage

Ist diese Analyse gut?

Nicht für kleine  $x$ , große  $x$

# Genauere Analyse ALGONE

```
algOne(x: ℕ, y: ℕ) // Annahme: initial  $x \leq y$   
  if  $x = 0 \vee x = y$  then  
    return 1  
  return algOne(x-1, y-1) + algOne(x, y-1)
```

## Schritt 1

- Kosten ausrechnen...

# Genauere Analyse ALGONE

**algOne**( $x: \mathbb{N}, y: \mathbb{N}$ ) // Annahme: initial  $x \leq y$

So  
■

$x,y$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2		1	3	6	10	15	21	28	36	45	55	66	78	91	105
3			1	4	10	20	35	56	84	120	165	220	286	364	455
4				1	5	15	35	70	126	210	330	495	715	1001	1365
5					1	6	21	56	126	252	462	792	1287	2002	3003
6						1	7	28	84	210	462	924	1716	3003	5005
7							1	8	36	120	330	792	1716	3432	6435
8								1	9	45	165	495	1287	3003	6435
9									1	10	55	220	715	2002	5005
10										1	11	66	286	1001	3003
11											1	12	78	364	1365
12												1	13	91	455
13													1	14	105
14														1	15
15															1

# Genaue Analyse ALGONE

```
algOne(x:  $\mathbb{N}$ , y:  $\mathbb{N}$ ) // Annahme: initial  $x \leq y$   
  if  $x = 0 \vee x = y$  then  
    return 1  
  return algOne(x-1, y-1) + algOne(x, y-1)
```

## Schritt 1

- Kosten ausrechnen...
- Zahlen anstarren

# Genauere Analyse ALGONE

`algOne(x: ℕ, y: ℕ) // Annahme: initial  $x \leq y$`

So

■

■

x,y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2		1	3	6	10	15	21	28	36	45	55	66	78	91	105
3			1	4	10	20	35	56	84	120	165	220	286	364	455
4				1	5	15	35	70	126	210	330	495	715	1001	1365
5					1	6	21	56	126	252	462	792	1287	2002	3003
6						1	7	28	84	210	462	924	1716	3003	5005
7							1	8	36	120	330	792	1716	3432	6435
8								1	9	45	165	495	1287	3003	6435
9									1	10	55	220	715	2002	5005
10										1	11	66	286	1001	3003
11											1	12	78	364	1365
12												1	13	91	455
13													1	14	105
14														1	15
15															1

# Genaue Analyse ALGONE

algOne( $x: \mathbb{N}, y: \mathbb{N}$ ) // Annahme: initial  $x \leq y$

$x, y$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
2		1	3	6								66	78	91	105	
3			1	4								5	220	286	364	455
4												0	495	715	1001	1365
5												2	792	1287	2002	3003
6												2	924	1716	3003	5005
7												0	792	1716	3432	6435
8												5	495	1287	3003	6435
9												5	220	715	2002	5005
10												66	286	1001	3003	
11												12	78	364	1365	
12												1	13	91	455	
13													1	14	105	
14														1	15	
15															1	

So

■

■

The OEIS is supported by [the many generous donors to the OEIS Foundation](#).

0 1 3 6 2 7  
 : 13  
 : 20  
 23 12  
 10 22 11 21

THE ON-LINE ENCYCLOPEDIA OF INTEGER SEQUENCES®

founded in 1964 by N. J. A. Sloane

1, 5, 15, 35, 70, 126, 210, 330, 495, 715, 1001, 1365   [Hints](#)

(Greetings from [The On-Line Encyclopedia of Integer Sequences!](#))

Search: **seq:1,5,15,35,70,126,210,330,495,715,1001,1365**

Displaying 1-1 of 1 result found. page 1

Sort: [relevance](#) | [references](#) | [number](#) | [modified](#) | [created](#) Format: [long](#) | [short](#) | [data](#)

**A000332** Binomial coefficient binomial(n,4) = n\*(n-1)\*(n-2)\*(n-3)/24. +30  
360  
 (Formerly M3853 N1578)

0, 0, 0, 0, 1, 5, 15, 35, 70, 126, 210, 330, 495, 715, 1001, 1365, 1820, 2380, 3060, 3876, 4845, 5985, 7315, 8855, 10626, 12650, 14950, 17550, 20475, 23751, 27405, 31465, 35960, 40920, 46376, 52360, 58905, 66045, 73815, 82251, 91390, 101270, 111930, 123410 ([list](#); [graph](#); [refs](#); [listen](#); [history](#); [text](#); [internal format](#))

OFFSET 0,6

COMMENTS Number of intersection points of diagonals of convex n-gon where no more than two diagonals intersect at any point in the interior.  
 Also the number of equilateral triangles with vertices in an equilateral triangular array of points with n rows (offset 1), with any orientation. - [Ignacio Larrosa Cañestro](#), Apr 09 2002. [See [Les Reid](#) link for proof. - [N. J. A. Sloane](#), Apr 02 2016]  
 Start from cubane and attach amino acids according to the reaction scheme that describes the reaction between the active sites. See the hyperlink on chemistry. - [Robert G. Wilson v](#), Aug 02 2002  
 For n>0, a(n) = (-1/8)\*(coefficient of x in Zagier's polynomial P(2n,n)). (Zagier's polynomials are used by PARI/GP for acceleration of alternating or positive series.)  
 Figurate numbers based on the 4-dimensional regular convex polytope called the regular 4-simplex, pentachoron, 5-cell, pentaptope or 4-hypertetrahedron with Schläfli symbol {3,3,3}. a(n) = ((n\*(n-1)\*(n-2)\*(n-3))/4!). - [Michael J. Welch](#) (mjwl(AT)ntlworld.com), Apr 01 2004, [R. J. Mathar](#), Jul 07 2009  
 Maximal number of crossings that can be created by connecting n vertices with straight lines. [Amir D. Faris](#) (redsell(AT)univulh.ca), Jan 30 2019  
 (ahmedfares(AT)my.deja.com), Feb 21 2001

Search: **seq:1,3,6,10,15,21,28,3**

Displaying 1-10 of 20 results found

Sort: [relevance](#) | [references](#) | [number](#)

**A000217** Triangular numbers:  
 + n.  
 (Formerly M2535 N1002)

0, 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 66, 78, 91, 105, 120, 136, 153, 171, 190, 210, 231, 253, 276, 300, 325, 351, 378, 407, 437, 468, 500, 533, 567, 603, 640, 679, 719, 761, 805, 851, 899, 949, 1001, 1055, 1111, 1169, 1229, 1291, 1355, 1421, 1489, 1559, 1631, 1705, 1781, 1859, 1939, 2021, 2105, 2191, 2279, 2369, 2461, 2555, 2651, 2749, 2849, 2951, 3055, 3161, 3269, 3379, 3491, 3605, 3721, 3839, 3959, 4081, 4205, 4331, 4459, 4589, 4721, 4855, 4991, 5129, 5269, 5411, 5555, 5701, 5849, 5999, 6151, 6305, 6461, 6619, 6779, 6941, 7105, 7271, 7439, 7609, 7781, 7955, 8131, 8309, 8489, 8671, 8855, 9041, 9229, 9419, 9611, 9805, 10001, 10201, 10403, 10607, 10813, 11021, 11231, 11443, 11657, 11873, 12091, 12311, 12533, 12757, 12983, 13211, 13441, 13673, 13907, 14143, 14381, 14621, 14863, 15107, 15353, 15601, 15851, 16103, 16357, 16613, 16871, 17131, 17393, 17657, 17923, 18191, 18461, 18733, 19007, 19283, 19561, 19841, 20123, 20407, 20693, 20981, 21271, 21563, 21857, 22153, 22451, 22751, 23053, 23357, 23663, 23971, 24281, 24593, 24907, 25223, 25541, 25861, 26183, 26507, 26833, 27161, 27491, 27823, 28157, 28493, 28831, 29171, 29513, 29857, 30203, 30551, 30901, 31253, 31607, 31963, 32321, 32681, 33043, 33407, 33773, 34141, 34511, 34883, 35257, 35633, 36011, 36391, 36773, 37157, 37543, 37931, 38321, 38713, 39107, 39503, 39901, 40301, 40703, 41107, 41513, 41921, 42331, 42743, 43157, 43573, 43991, 44411, 44833, 45257, 45683, 46111, 46541, 46973, 47407, 47843, 48281, 48721, 49163, 49607, 50053, 50501, 50951, 51403, 51857, 52313, 52771, 53231, 53693, 54157, 54623, 55091, 55561, 56033, 56507, 56983, 57461, 57941, 58423, 58907, 59393, 59881, 60371, 60863, 61357, 61853, 62351, 62851, 63353, 63857, 64363, 64871, 65381, 65893, 66407, 66923, 67441, 67961, 68483, 69007, 69533, 70061, 70591, 71123, 71657, 72193, 72731, 73271, 73813, 74357, 74903, 75451, 76001, 76553, 77107, 77663, 78221, 78781, 79343, 79907, 80473, 81041, 81611, 82183, 82757, 83333, 83911, 84491, 85073, 85657, 86243, 86831, 87421, 88013, 88607, 89203, 89801, 90401, 91003, 91607, 92213, 92821, 93431, 94043, 94657, 95273, 95891, 96511, 97133, 97757, 98383, 99011, 99641, 100273, 100907, 101543, 102181, 102821, 103463, 104107, 104753, 105401, 106051, 106703, 107357, 108013, 108671, 109331, 110093, 110857, 111623, 112391, 113161, 113933, 114707, 115483, 116261, 117041, 117823, 118607, 119393, 120181, 120971, 121763, 122557, 123353, 124151, 124951, 125753, 126557, 127363, 128171, 128981, 129793, 130607, 131423, 132241, 133061, 133883, 134707, 135533, 136361, 137191, 138023, 138857, 139693, 140531, 141371, 142213, 143057, 143903, 144751, 145601, 146453, 147307, 148163, 149021, 149881, 150743, 151607, 152473, 153341, 154211, 155083, 155957, 156833, 157711, 158591, 159473, 160357, 161243, 162131, 163021, 163913, 164807, 165703, 166601, 167501, 168403, 169307, 170213, 171121, 172031, 172943, 173857, 174773, 175691, 176611, 177533, 178457, 179383, 180311, 181241, 182173, 183107, 184043, 184981, 185921, 186863, 187807, 188753, 189701, 190651, 191603, 192557, 193513, 194471, 195431, 196393, 197357, 198323, 199291, 200261, 201233, 202207, 203183, 204161, 205141, 206123, 207107, 208093, 209081, 210071, 211063, 212057, 213053, 214051, 215051, 216053, 217057, 218063, 219071, 220081, 221093, 222107, 223123, 224141, 225161, 226183, 227207, 228233, 229261, 230291, 231323, 232357, 233393, 234431, 235471, 236513, 237557, 238603, 239651, 240701, 241753, 242807, 243863, 244921, 245981, 247043, 248107, 249173, 250241, 251311, 252383, 253457, 254533, 255611, 256691, 257773, 258857, 259943, 261031, 262121, 263213, 264307, 265403, 266501, 267601, 268703, 269807, 270913, 272021, 273131, 274243, 275357, 276473, 277591, 278711, 279833, 280957, 282083, 283211, 284341, 285473, 286607, 287743, 288881, 290021, 291163, 292307, 293453, 294601, 295751, 296903, 298057, 299213, 300371, 301531, 302693, 303857, 305023, 306191, 307361, 308533, 309707, 310883, 312061, 313241, 314423, 315607, 316793, 317981, 319171, 320363, 321557, 322753, 323951, 325151, 326353, 327557, 328763, 329971, 331181, 332393, 333607, 334823, 336041, 337261, 338483, 339707, 340933, 342161, 343391, 344623, 345857, 347093, 348331, 349571, 350813, 352057, 353303, 354551, 355801, 357053, 358307, 359563, 360821, 362081, 363343, 364607, 365873, 367141, 368411, 369683, 370957, 372233, 373511, 374791, 376073, 377357, 378643, 379931, 381221, 382513, 383807, 385103, 386401, 387701, 389003, 390307, 391613, 392921, 394231, 395543, 396857, 398173, 399491, 400811, 402133, 403457, 404783, 406111, 407441, 408773, 410107, 411443, 412781, 414121, 415463, 416807, 418153, 419501, 420851, 422203, 423557, 424913, 426271, 427631, 428993, 430357, 431723, 433091, 434461, 435833, 437207, 438583, 439961, 441341, 442723, 444107, 445493, 446881, 448271, 449663, 451057, 452453, 453851, 455251, 456653, 458057, 459463, 460871, 462281, 463693, 465107, 466523, 467941, 469361, 470783, 472207, 473633, 475061, 476491, 477923, 479357, 480793, 482231, 483671, 485113, 486557, 488003, 489451, 490901, 492353, 493807, 495263, 496721, 498181, 499643, 501107, 502573, 504041, 505511, 506983, 508457, 509933, 511411, 512891, 514373, 515857, 517343, 518831, 520321, 521813, 523307, 524803, 526301, 527801, 529303, 530807, 532313, 533821, 535331, 536843, 538357, 539873, 541391, 542911, 544433, 545957, 547483, 549011, 550541, 552073, 553607, 555143, 556681, 558221, 559763, 561307, 562853, 564401, 565951, 567503, 569057, 570613, 572171, 573731, 575293, 576857, 578423, 579991, 581561, 583133, 584707, 586283, 587861, 589441, 591023, 592607, 594193, 595781, 597371, 598963, 600557, 602153, 603751, 605351, 606953, 608557, 610163, 611771, 613381, 614993, 616607, 618223, 619841, 621461, 623083, 624707, 626333, 627961, 629591, 631223, 632857, 634493, 636131, 637771, 639413, 641057, 642703, 644351, 645991, 647633, 649277, 650923, 652571, 654221, 655873, 657527, 659183, 660841, 662501, 664163, 665827, 667493, 669161, 670831, 672503, 674177, 675853, 677531, 679211, 680893, 682577, 684263, 685951, 687641, 689333, 691027, 692723, 694421, 696121, 697823, 699527, 701233, 702941, 704651, 706363, 708077, 709793, 711511, 713231, 714953, 716677, 718403, 720131, 721861, 723593, 725327, 727063, 728801, 730541, 732283, 734027, 735773, 737521, 739271, 741023, 742777, 744533, 746291, 748051, 749813, 751577, 753343, 755111, 756881, 758653, 760427, 762203, 763981, 765761, 767543, 769327, 771113, 772901, 774691, 776483, 778277, 780073, 781871, 783671, 785473, 787277, 789083, 790891, 792701, 794513, 796327, 798143, 799961, 801781, 803603, 805427, 807253, 809081, 810911, 812743, 814577, 816413, 818251, 820091, 821933, 823777, 825623, 827471, 829321, 831173, 833027, 834883, 836741, 838601, 840463, 842327, 844193, 846061, 847931, 849803, 851677, 853553, 855431, 857311, 859193, 861077, 862963, 864851, 866741, 868633, 870527, 872423, 874321, 876221, 878123, 880027, 881933, 883841, 885751, 887663, 889577, 891493, 893411, 895331, 897253, 899177, 901103, 903031, 904961, 906893, 908827, 910763, 912701, 914641, 916583, 918527, 920473, 922421, 924371, 926323, 928277, 930233, 932191, 934151, 936113, 938077, 940043, 942011, 943981, 945953, 947927, 949903, 951881, 953861, 955843, 957827, 959813, 961801, 963791, 965783, 967777, 969773, 971771, 973771, 975773, 977777, 979783, 981791, 983801, 985813, 987827, 989843, 991861, 993881, 995903, 997927, 999953, 1001981, 1004011, 1006043, 1008077, 1010113, 1012151, 1014191, 1016233, 1018277, 1020323, 1022371, 1024421, 1026473, 1028527, 1030583, 1032641, 1034701, 1036763, 1038827, 1040893, 1042961, 1045031, 1047103, 1049177, 1051253, 1053331, 1055411, 1057493, 1059577, 1061663, 1063751, 1065841, 1067933, 1070027, 1072123, 1074221, 1076321, 1078423, 1080527, 1082633, 1084741, 1086851, 1088963, 1091077, 1093193, 1095311, 1097431, 1099553, 1101677, 1103803, 1105931, 1108061, 1110193, 1112327, 1114463, 1116601, 1118741, 1120883, 1123027, 1125173, 1127321, 1129471, 1131623, 1133777, 1135933, 1138091, 1140251, 1142413, 1144577, 1146743, 1148911, 1151081, 1153253, 1155427, 1157603, 1159781, 1161961, 1164143, 1166327, 1168513, 1170701, 1172891, 1175083, 1177277, 1179473, 1181671, 1183871, 1186073, 1188277, 1190483, 1192691, 1194901, 1197113, 1199327, 1201543, 1203761, 1205981, 1208203, 1210427, 1212653, 1214881, 1217111, 1219343, 1221577, 1223813, 1226051, 1228291, 1230533, 1232777, 1235023, 1237271, 1239521, 1241773, 1244027, 1246283, 1248541, 1250801, 1253063, 1255327, 1257593, 1259861, 1262131, 1264403, 1266677, 1268953, 1271231, 1273511, 1275793, 1278077, 1280363, 1282651, 1284941, 1287233, 1289527, 1291823, 1294121, 1296421, 1298723, 1301027, 1303333, 1305641, 1307951, 1310263, 1312577, 1314893, 1317211

# Genauere Analyse ALGONE

```
algOne(x: ℕ, y: ℕ) // Annahme: initial  $x \leq y$   
  if  $x = 0 \vee x = y$  then  
    return 1  
  return algOne(x-1, y-1) + algOne(x, y-1)
```

## Schritt 1

- Kosten ausrechnen...
- Zahlen anstarren

## Schritt 2

- Vermutung bilden:
  - $T(x, y) \leq 2\binom{y}{x} - 1$
- Vermutung beweisen

# Genauere Analyse ALGONE

```

algOne(x: ℕ, y: ℕ) // Annahme: initial  $x \leq y$ 
  if  $x = 0 \vee x = y$  then
    return 1
  return algOne(x-1, y-1) + algOne(x, y-1)
  
```

## Schritt 1

- Kosten ausrechnen...
- Zahlen anstarren

## Schritt 2

- Vermutung bilden:
  - $T(x, y) \leq 2 \binom{y}{x} - 1$
- Vermutung beweisen

## Beweis per Induktion

- für  $i \geq 0$ :  $T(i, i) = 2 \binom{i}{i} - 1 = 2 - 1 = 1$
- Ang.: Vermutung gilt für alle kleineren Werte von  $x$  und  $y$

$$\begin{aligned}
 T(x, y) &= T(x-1, y-1) + T(x, y-1) + 1 \\
 &= 2 \binom{y-1}{x-1} - 1 + 2 \binom{y-1}{x} - 1 + 1 \\
 &= 2 \left( \binom{y-1}{x-1} + \binom{y-1}{x} \right) - 1 \\
 &= 2 \binom{y}{x} - 1
 \end{aligned}$$

# Genauere Analyse ALGONE

**algOne**( $x: \mathbb{N}, y: \mathbb{N}$ ) // Annahme: initial  $x \leq y$

**if**  $x = 0 \vee x = y$  **then**

Eigenschaften [ Bearbeiten | Quelltext bearbeiten ]

Wird außer  $k$  auch  $n$  auf nichtnegative ganze Zahlen eingeschränkt, so gilt:

- $\binom{n}{k}$  ist stets eine nichtnegative ganze Zahl. Ist  $k \leq n$ , so ist  $\binom{n}{k} \geq 1$ , anderenfalls ist  $\binom{n}{k} = 0$ .

So

- $\binom{n}{0} = 1 = \binom{n}{n}$

- $\binom{n}{1} = n = \binom{n}{n-1}$

- $\binom{n}{k} = \frac{n-k+1}{k} \binom{n}{k-1}$

- $\binom{n}{k} = \frac{n}{k} \cdot \binom{n-1}{k-1} \Leftrightarrow k \cdot \binom{n}{k} = n \cdot \binom{n-1}{k-1}$

- $\binom{n+1}{k+1} = \binom{n}{k} + \binom{n}{k+1}$ . Für  $n = k$  ist der rechte Summand  $\binom{n}{k+1} = 0$ .

- $T(x, y) \leq 2 \binom{y}{x} - 1$

- Vermutung beweisen

## Beweis per Induktion

- für  $i \geq 0$ :  $T(i, i) = 2 \binom{i}{i} - 1 = 2 - 1 = 1$

g.: Vermutung gilt für alle kleineren Werte von  $x$  und  $y$

$$T(x, y) = T(x-1, y-1) + T(x, y-1) + 1$$

$$= 2 \binom{y-1}{x-1} - 1 + 2 \binom{y-1}{x} - 1 + 1$$

$$= 2 \left( \binom{y-1}{x-1} + \binom{y-1}{x} \right) - 1$$

$$= 2 \binom{y}{x} - 1$$

# Genauere Analyse ALGONE

```

algOne(x: ℕ, y: ℕ) // Annahme: initial  $x \leq y$ 
  if  $x = 0 \vee x = y$  then
    return 1
  return algOne(x-1, y-1) + algOne(x, y-1)
  
```

## Schritt 1

- Kosten ausrechnen...
- Zahlen anstarren

## Schritt 2

- Vermutung bilden:
  - $T(x, y) \leq 2 \binom{y}{x} - 1$
- Vermutung beweisen

## Beweis per Induktion

- für  $i \geq 0$ :  $T(i, i) = 2 \binom{i}{i} - 1 = 2 - 1 = 1$
- Ang.: Vermutung gilt für alle kleineren Werte von  $x$  und  $y$

$$\begin{aligned}
 T(x, y) &= T(x-1, y-1) + T(x, y-1) + 1 \\
 &= 2 \binom{y}{x} - 1
 \end{aligned}$$

# Genauere Analyse ALGONE

```

algOne(x: ℕ, y: ℕ) // Annahme: initial  $x \leq y$ 
  if  $x = 0 \vee x = y$  then
    return 1
  return algOne(x-1, y-1) + algOne(x, y-1)
  
```

## Schritt 1

- Kosten ausrechnen...
- Zahlen anstarren

## Schritt 2

- Vermutung bilden:
  - $T(x, y) \leq 2 \binom{y}{x} - 1$
- Vermutung beweisen

## Beweis per Induktion

- für  $i \geq 0$ :  $T(i, i) = 2 \binom{i}{i} - 1 = 2 - 1 = 1$
- Ang.: Vermutung gilt für alle kleineren Werte von  $x$  und  $y$

$$\begin{aligned}
 T(x, y) &= T(x-1, y-1) + T(x, y-1) + 1 \\
 &= 2 \binom{y}{x} - 1
 \end{aligned}$$

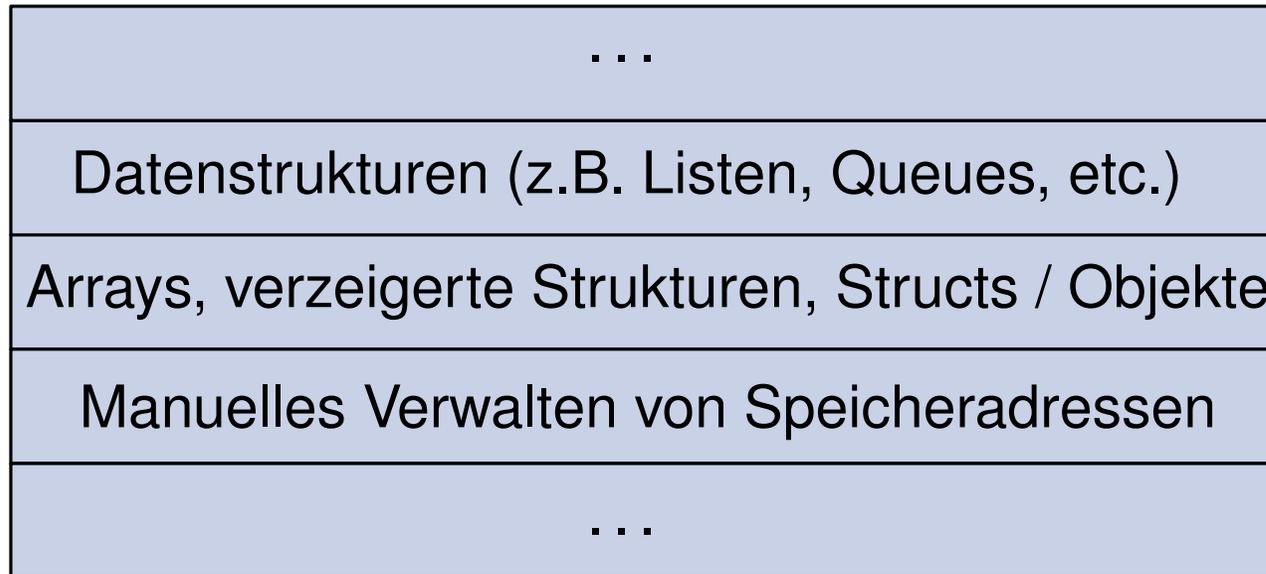
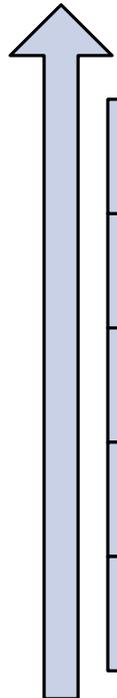
- Moment mal...
  - $\binom{y}{x} = \binom{y-1}{x-1} + \binom{y-1}{x} \dots ?$
  - Das ist doch genau **algOne**!
  - $T(x, y) \in \Theta\left(\binom{y}{x}\right)$

:-)

# Motivation: Datenstrukturen

Abstraktionsebenen auf dem Speicher

hoch



**Bsp: Queue**

■ **pushBack(e)**  $O(1)$

■ **popFront()**  $O(1)$

■ **size()**  $O(1)$

*z.B. mittels verketteter Liste*

tief

# Motivation: Amortisierte Analyse

- Laufzeit von Operationen auf DS nicht immer gleich
  - z.B.: Operation meist günstig, seltener teurer

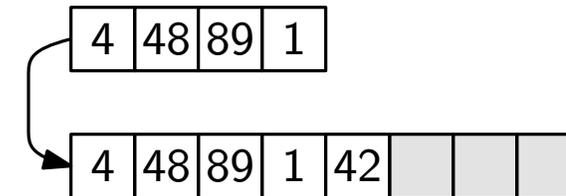
## Idee

- **amortisierte Kosten**: durchschnittliche Kosten pro Operation in einer Folge von Operationen

## Verschiedene Techniken zur Analyse

- Aggregation
- Charging
- Konto
- Potential

Beispiel: unbeschränktes Array



# Beispiel: Binärzähler

## Idee

- Array  $A$  verwaltet Bits
- Funktion **increment()** erhöht Zähler

## Algorithmus

- suche Stelle  $i$  mit rechtester 0
- setze Stelle  $i$  auf 1
- setze Stellen rechts von  $i$  auf 0

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

0	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

0	0	0	0	0	1	1	0
---	---	---	---	---	---	---	---

0	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---

0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

...

# Beispiel: Binärzähler

## Idee

- Array  $A$  verwaltet Bits
- Funktion **increment()** erhöht Zähler

## Algorithmus in Pseudocode

### increment()

```

  i := 0
  while A[i] = 1 do
    A[i] := 0
    i := i+1
  A[i] := 1

```

## Laufzeit?

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

0	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

0	0	0	0	0	1	1	0
---	---	---	---	---	---	---	---

0	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---

0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

...

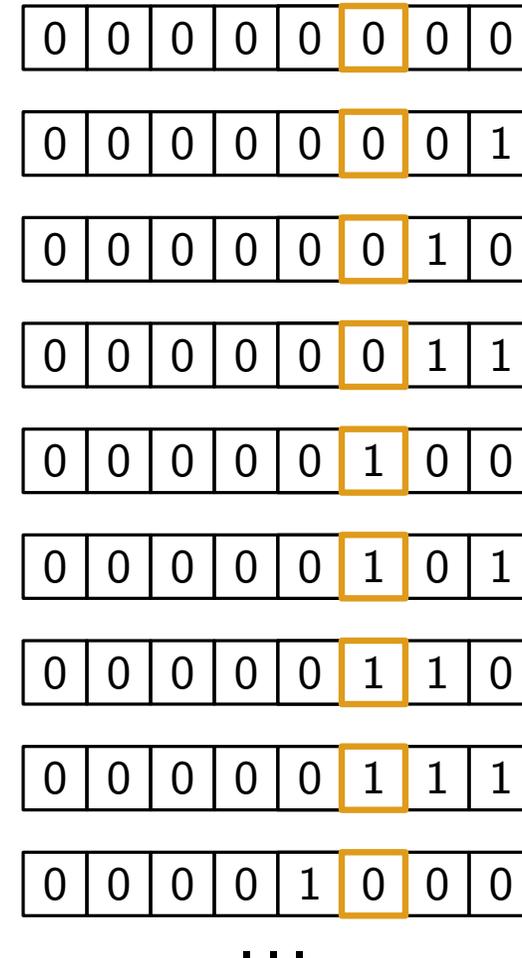
# Analyse: Aggregatmethode

## Idee

- berechne Gesamtkosten
- teile durch Anzahl von Operationen

## Beobachtung

- nicht jedes Bit flipt bei jeder Operation
- $A[0]$  flipt jedes mal,  $A[1]$  jedes zweite, etc.
- $A[i]$  flipt bei jedem  $2^i$ -ten Inkrement
- Bei  $n$  Aufrufen:  $A[i]$  flipt  $\lfloor \frac{n}{2^i} \rfloor$  mal



# Analyse: Aggregatmethode

## Beobachtung

- nicht jedes Bit flipt bei jeder Operation
- $A[0]$  flipt jedes mal,  $A[1]$  jedes zweite, etc.
- $A[i]$  flipt bei jedem  $2^i$ -ten Inkrement
- Bei  $n$  Aufrufen:  $A[i]$  flipt  $\lfloor \frac{n}{2^i} \rfloor$  mal

In Summe ergibt sich für  $k$  bits:

$$\sum_{i=0}^{k-1} \lfloor \frac{n}{2^i} \rfloor < n \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n$$

D.h.: pro Operation nur 2 Bit-Flips

Somit: **amortisierte Kosten** in  $O(1)$

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

0	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

0	0	0	0	0	1	1	0
---	---	---	---	---	---	---	---

0	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---

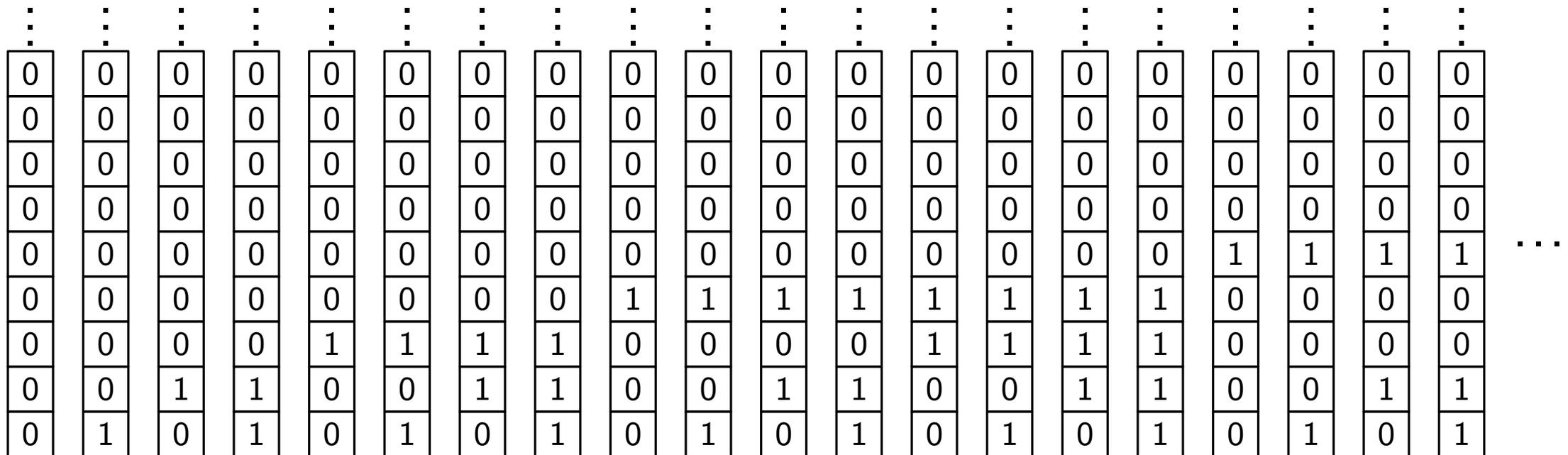
0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

...

# Analyse mittels Charging

## Idee

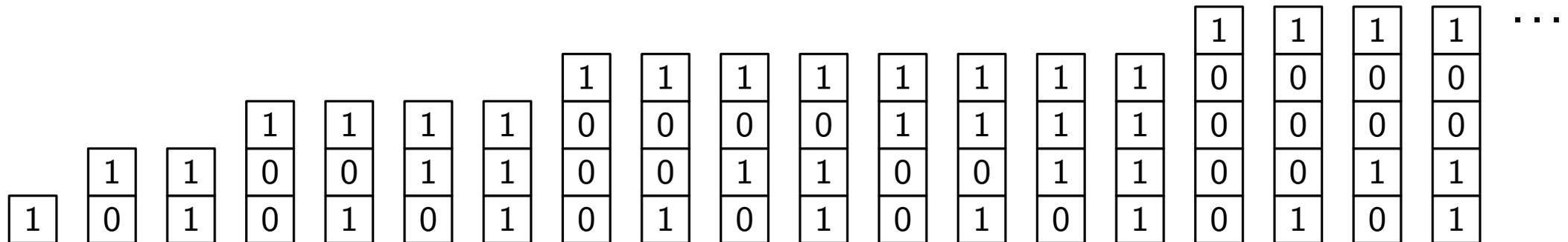
- teure Operationen legen Kosten um auf günstige Operationen



# Analyse mittels Charging

## Idee

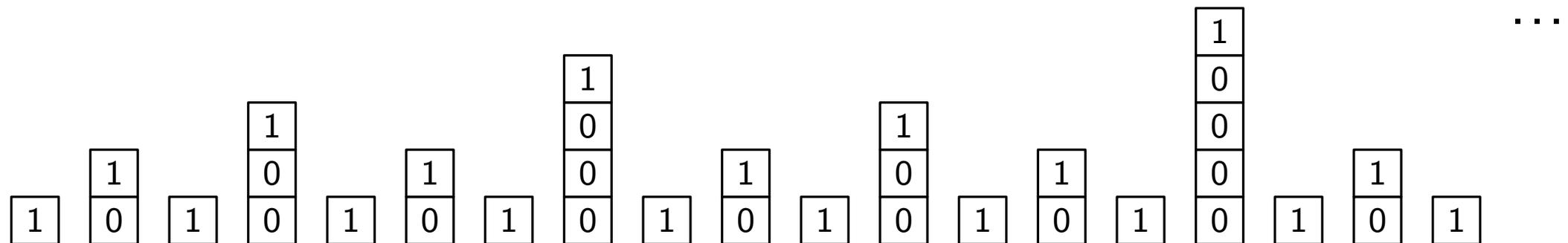
- teure Operationen legen Kosten um auf günstige Operationen



# Analyse mittels Charging

## Idee

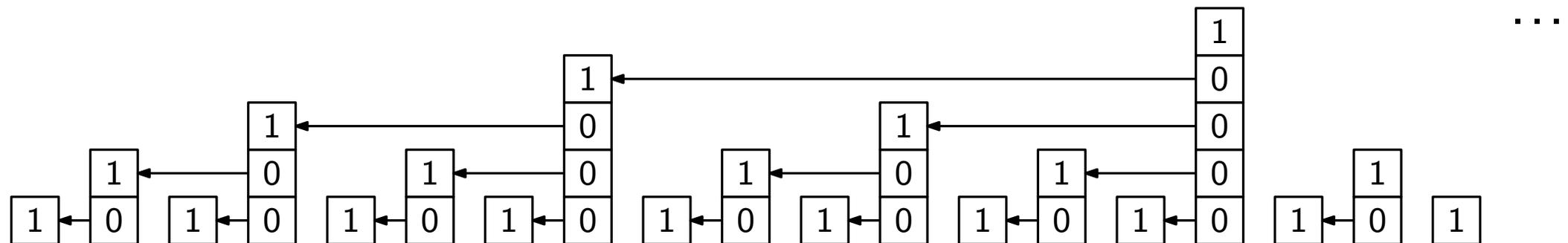
- teure Operationen legen Kosten um auf günstige Operationen



# Analyse mittels Charging

## Idee

- teure Operationen legen Kosten um auf günstige Operationen





# Analyse: Potentialmethode

## Idee

- definiere Potentialfunktion  $\Phi(D_i)$ 
  - Maß für Unaufgeräumtheit von  $D$  zum Zeitpunkt  $i$
- definiere amortisierte Kosten  $\hat{c}_i = c_i + \underbrace{\Phi(D_i) - \Phi(D_{i-1})}_{\text{Anstieg des Potenzials}}$ 
  - tatsächliche Kosten

# Analyse: Potentialmethode

## Idee

- definiere Potentialfunktion  $\Phi(D_i)$ 
  - Maß für Unaufgeräumtheit von  $D$  zum Zeitpunkt  $i$
- definiere amortisierte Kosten  $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$

## Definition sinnvoll?

$$\begin{aligned}
 \sum_{i=1}^n \hat{c}_i &= \sum_{i=1}^n (c_i + \Phi(D_i) - \Phi(D_{i-1})) = \sum_{i=1}^n (c_i) + \sum_{i=1}^n (\Phi(D_i) - \Phi(D_{i-1})) \\
 &= \sum_{i=1}^n (c_i) + \Phi(D_n) - \Phi(D_0)
 \end{aligned}$$

Falls  $\Phi(D_n) \geq \Phi(D_0)$ : amortisierte Kosten obere Schranke für tatsächliche Kosten

$\Rightarrow$  fordere  $\Phi(D_i) \geq \Phi(D_0)$  f.a.  $i > 0$     oder:  $\Phi(D_0) = 0$  und  $\Phi(D_i) \geq 0$  f.a.  $i > 0$

# Analyse: Potentialmethode

Potentialfunktion für Binärzähler:

- def.  $\Phi(D_i) = \sum_{j=0}^{k-1} A[j]$  (Anzahl 1-bits zum Zeitpunkt  $i$ )

Dann ergibt sich:

- $\Phi(D_0) = 0, \Phi(D_i) \geq 0$  f.a.  $i > 0$  *gültige Pot.fun. :)*
- tatsächliche Kosten:  $c_i = \#01\text{-flips} + \#10\text{-flips}$
- Potentialänderung:  $\#01\text{-flips} - \#10\text{-flips}$
- amort. Kosten:  $\hat{c}_i = 2 \cdot \#01\text{-flips} \leq 2$

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

0	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

0	0	0	0	0	1	1	0
---	---	---	---	---	---	---	---

0	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---

0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

...

**Definition:** amortisierte Kosten  $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$

# Beispiel 2: Stacks und Queues

## Wiederholung

Queue

- **push**(e)  $O(1)$
- **pop**()  $O(1)$
- **size**()  $O(1)$

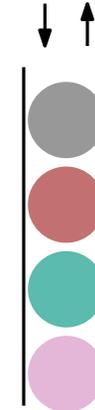
*z.B. mittels verketteter Liste*



Stack

- **push**(e)  $O(1)$
- **pop**()  $O(1)$
- **size**()  $O(1)$

*z.B. mittels verketteter Liste*



## Frage

Angenommen wir können Stacks (als Blackbox) verwenden.  
 (Wie) können wir daraus eine Queue bauen?

# Eine Queue aus Stacks

## Algorithmische Umsetzung

- **push**: auf  $A$  pushen
- **pop**:
  - falls  $B$  voll: von  $B$  poppen
  - falls  $B$  leer: alles von  $A$  nach  $B$  verschieben

$O(1)$

$O(|A|)$

$O(1)$

$O(|A|)$

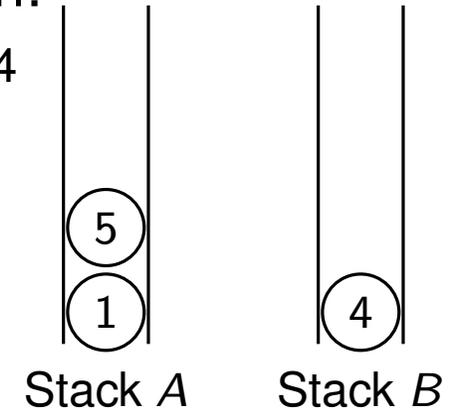
Operationen:

■ **push**: 3, 1, 4

■ **pop**()

■ **pop**()

■ **push**: 1,5



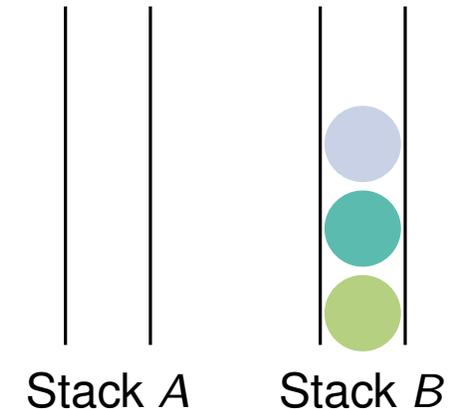
# Analyse: Kontomethode

## Idee

- günstige Operationen bauen Guthaben auf
- teure Operationen nutzen Guthaben

## Umsetzung hier

- Gesamtkosten linear in **push** und **pop** Aufrufen
- Bei **push**: zahle 3
- Bei **pop**: zahle 1
  - Falls *B* leer: nutze 2 Guthaben für *A.pop()* und *B.push()* (für alle Elemente in *A*)
  - Falls *B* voll: *B.pop()* mit Kosten 1
- Konto wird nie negativ
- $3, 1 \in \Theta(1) \Rightarrow$  konstante amortisierte Kosten



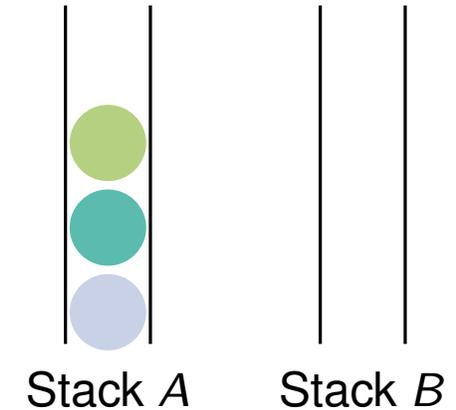
# Analyse: Potentialmethode

## Potentialfunktion

- Definiere  $\Phi(D_i) = 2 \cdot |A_i|$  (Anzahl Elemente auf  $A$  zum Zeitpunkt  $i$ )

## Analyse

- $\Phi(D_0) = 0, \Phi(D_i) \geq 0$  f.a.  $i > 0$  (d.h. gültige Potentialfunktion)
  - amortisierte Kosten **push**:
    - $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 3$
  - amortisierte Kosten **pop**:
    - $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1$
    - $c_i = 2 \cdot |A_{i-1}| + 1$
    - $\Phi(D_i) - \Phi(D_{i-1}) = -2|A_{i-1}|$
- } amortisierte Kosten in  $O(1)$



**Definition:** amortisierte Kosten  $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$