# テキスト処理 第13回 (2008-07-15)
# /a*aaaa/ での try 呼出し回数

田中哲
産業技術総合研究所
情報技術研究部門
akr@isc.senshu-u.ac.jp
http://staff.aist.go.jp/tanaka-akira/textprocess-2008/

# レポート

- a が n 個並んでいる文字列に /a*aaaa/ をマッチさせたときに try が呼び出される回数を n に対する関数として求めよ
- try_cat, try_alt は任意個引数版を使う

- ∠切 2008-07-15 12:00
- RENANDI

# /a*aaaa/ =~ "a" * n

- ある特定の n に対し、try の呼び出し回数は以下で求められる
- count_try(
  [:cat, [:rep, "a"], "a", "a", "a", "a"],
  "a" * n)

# 想定されるレポートの内容

- 求めた式 (関数)
- その式が求まった理由
  - グラフから求めるのではなく、プログラムの動作をたどって数える

# ヒント

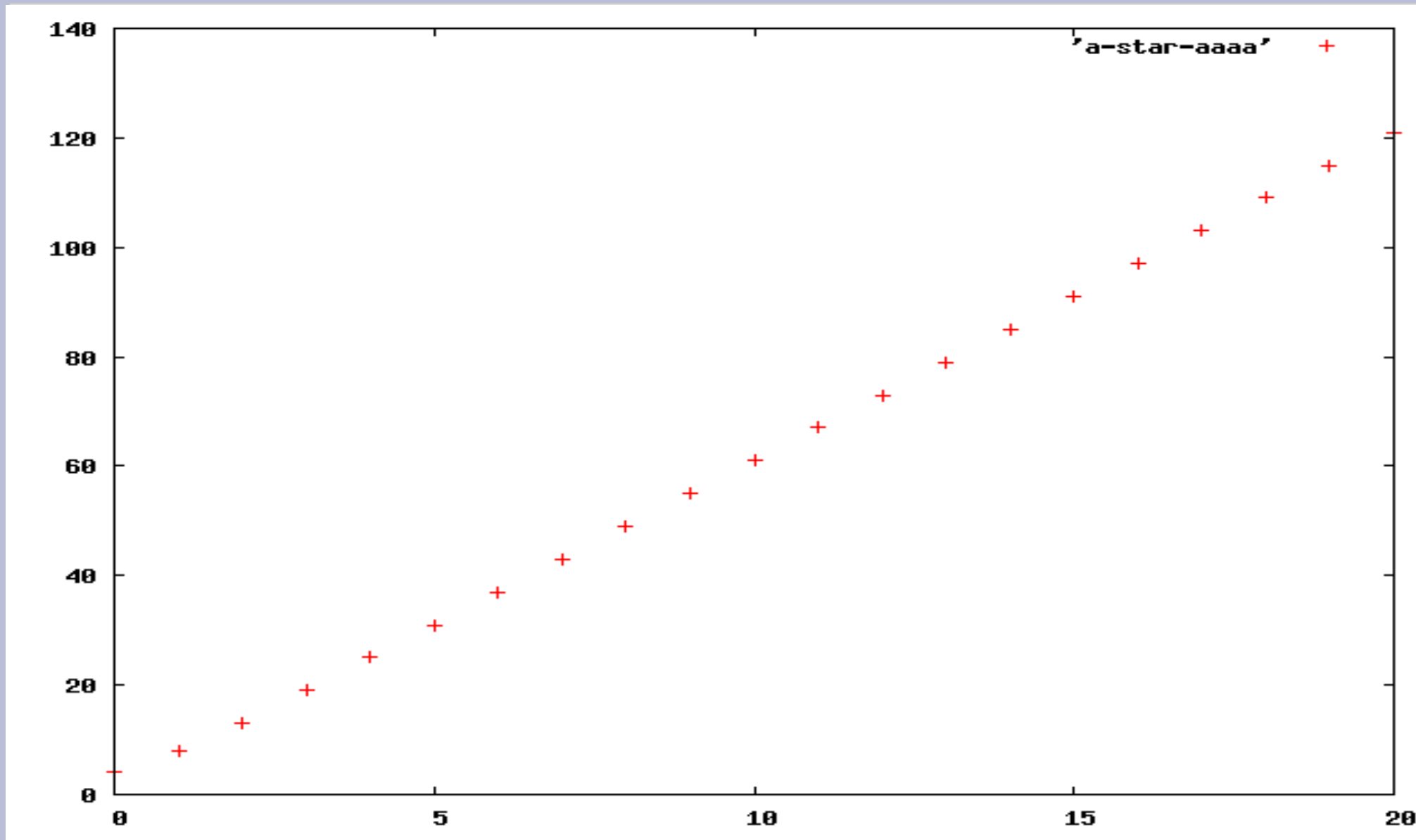
- わからなければ try の先頭に p exp とか p [pos, exp] とか入れて動作をたどる
- n が小さいときには場合分けが必要

# とりあえず測定

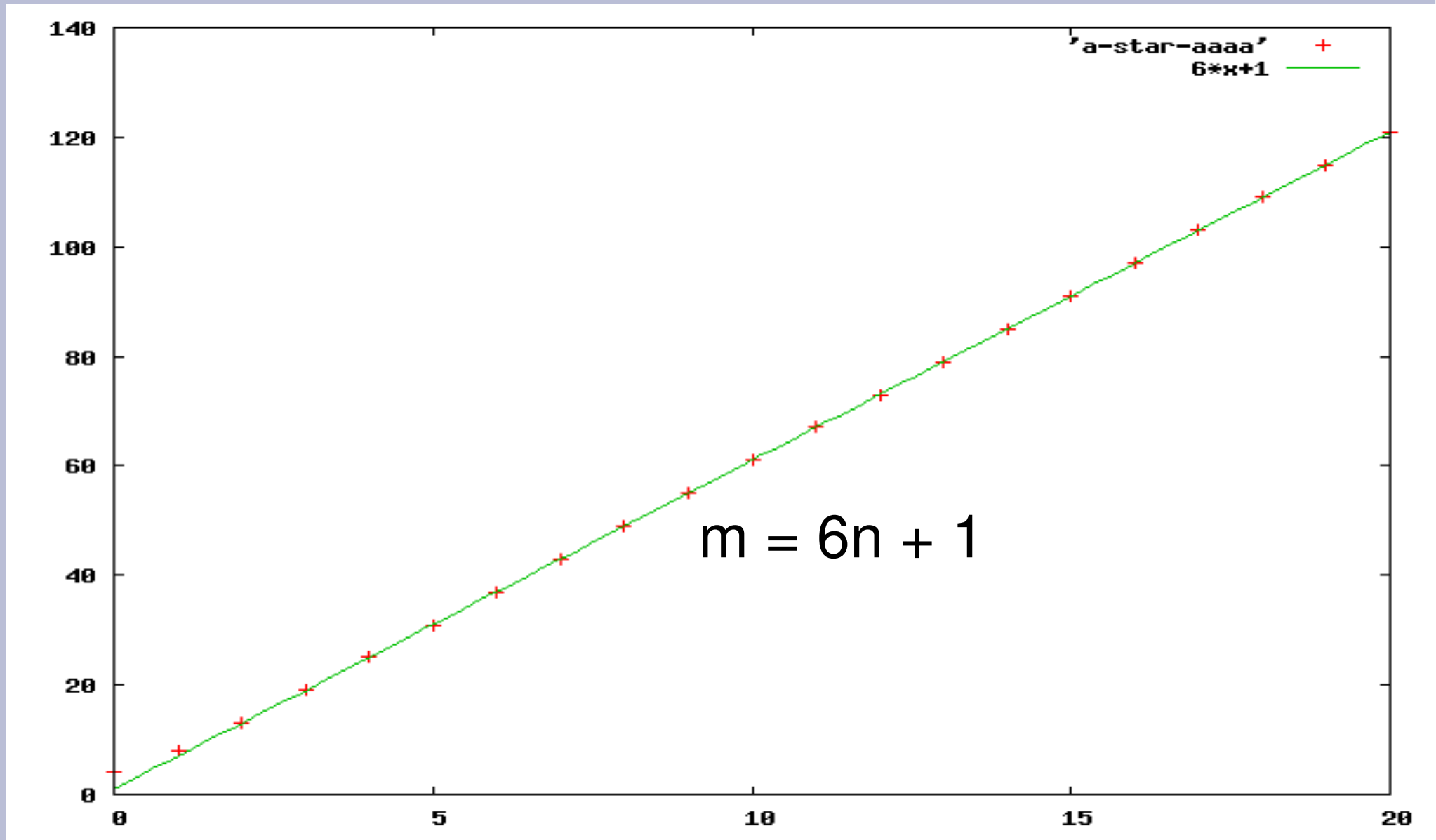
```
0.upto(20) {|n|
m = count_try([:cat, [:rep, "a"], "a", "a", "a", "a"],
                    "a" * n)
puts "#{n} #{m}"
}
```

| | | | | |
|---|---|---|---|---|
| 0 4 | 5 31 | 10 61 | 15 91 | 20 121 |
| 1 8 | 6 37 | 11 67 | 16 97 | |
| 2 13 | 7 43 | 12 73 | 17 103 | |
| 3 19 | 8 49 | 13 79 | 18 109 | |
| 4 25 | 9 55 | 14 85 | 19 115 | |

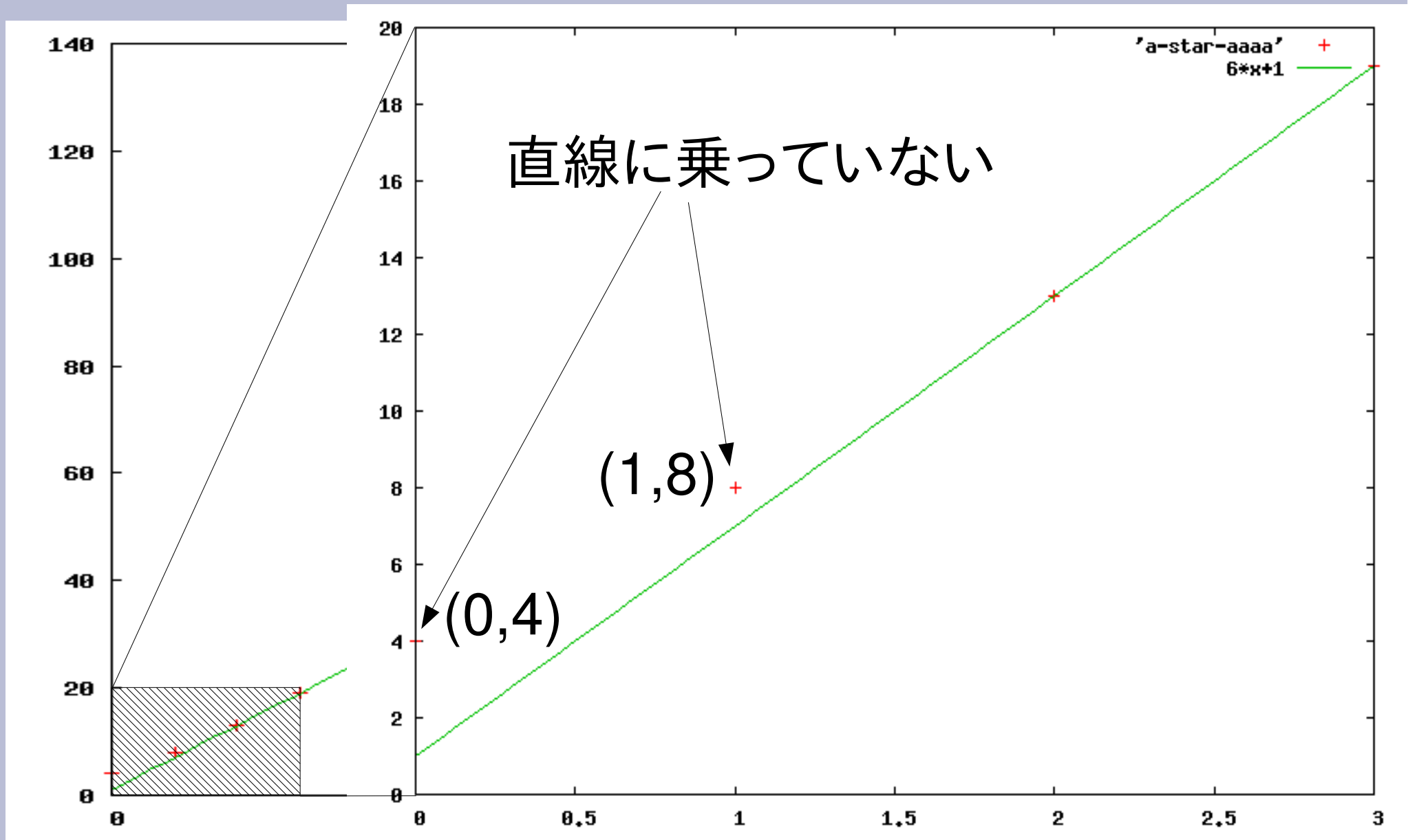
# グラフ



# 直線を引く



# 原点付近の拡大

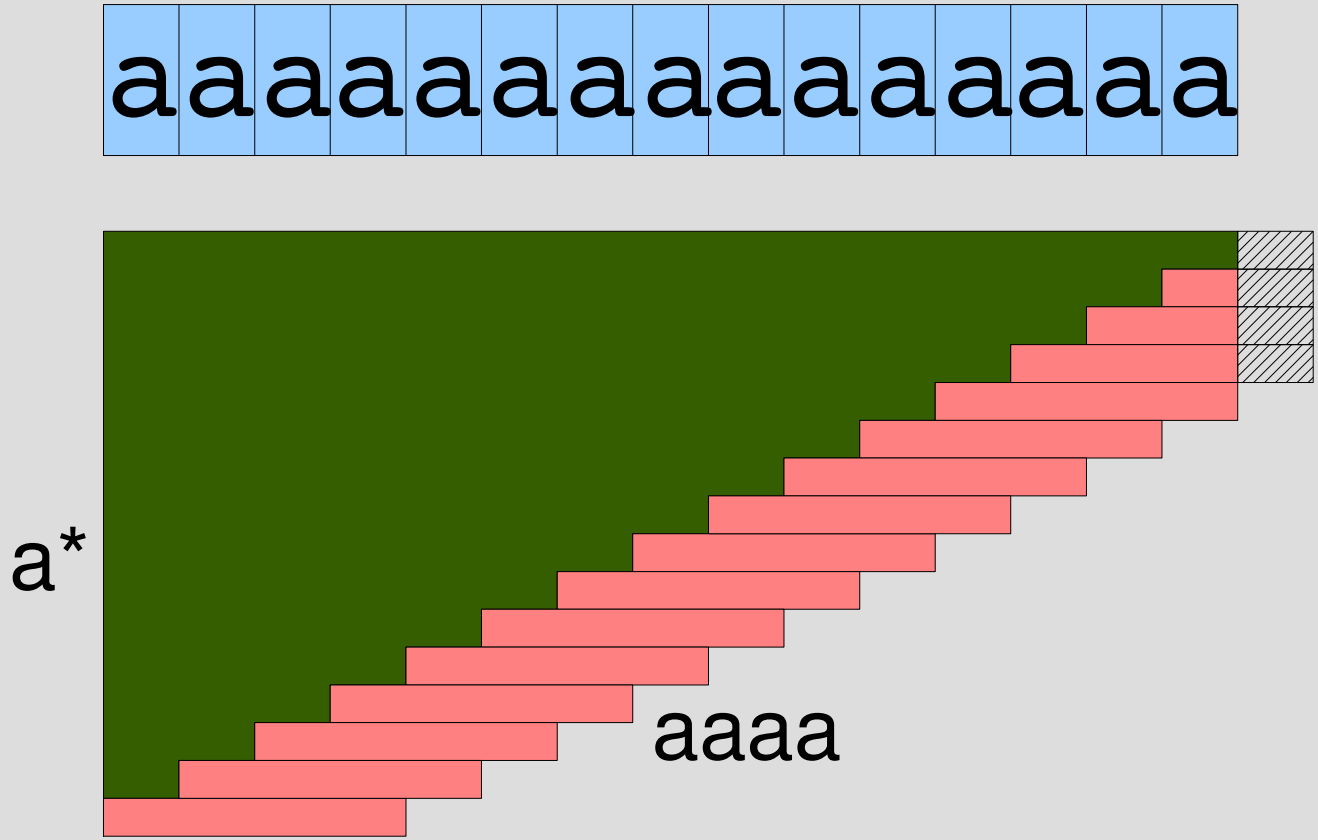

'a-star-aaaa'
6*x+1
直線に乗っていない
(1,8)
(0,4)
140
120
100
80
60
40
20
0
20
18
16
14
12
10
8
6
4
2
0
0
0.5
1
1.5
2
2.5
3

# グラフからみた結果

tryが呼び出される回数は

- n = 0 のとき 4回
- n = 1 のとき 8回
- それ以外のとき 6n+1 回

# /a*aaaa/ の処理の流れ



# try の呼出しを表示

- def try(re, str, pos, md, &b)
  p [pos, re]
  ...

# count_tr([...], "a"*4)を表示

[0, [:cat, [:rep, "a"], "a", "a", "a", "a"]]
[0, [:rep, "a"]]
[0, "a"]
[1, [:rep, "a"]]
[1, "a"]
[2, [:rep, "a"]]
[2, "a"]
[3, [:rep, "a"]]
[3, "a"]
[4, [:rep, "a"]]
[4, "a"]

a* が伸びていく

[4, "a"]
[3, "a"]
[4, "a"]
[2, "a"]
[3, "a"]
[4, "a"]

4から、
3から、
2から、
aaaaの
マッチ

[1, "a"]
[2, "a"]
[3, "a"]
[4, "a"]
[0, "a"]
[1, "a"]
[2, "a"]
[3, "a"]

1から、
0から、
aaaa の
マッチ

# count_tr([...], "a"*n)を表示 (n=4)

[0, [:cat, [:rep, "a"], "a", "a", "a", "a"]]
[0, [:rep, "a"]]
[0, "a"]
[1, [:rep, "a"]]
[1, "a"]
[2, [:rep, "a"]]
[2, "a"]
[3, [:rep, "a"]]
[3, "a"]
[4, [:rep, "a"]]
[4, "a"]

a* が伸びていく

1+2(n+1)回

[4, "a"]
[3, "a"]
[4, "a"]
[2, "a"]
[3, "a"]
[4, "a"]

nから、
n-1から、
n-2から。

6回

[1, "a"]
[2, "a"]
[3, "a"]
[4, "a"]
[0, "a"]
[1, "a"]
[2, "a"]
[3, "a"]

n-3から、
…
0から。

4(n-2)回

# 総計 (nが大きいとき)

- $1+2(n+1)+6+4(n-2)=6n+1$

# nが小さいとき

- n が 2より小さいと 4(n-2) が負になっておかしくなる

[0, [:cat, [:rep, "a"], "a", "a", "a", "a"]]
[0, [:rep, "a"]]
[0, "a"]
[1, [:rep, "a"]]
[1, "a"]
[2, [:rep, "a"]]
[2, "a"]
[3, [:rep, "a"]]
[3, "a"]
[4, [:rep, "a"]]
[4, "a"]

a* が伸びていく

1+2(n+1)回

[4, "a"]
[3, "a"]
[4, "a"]
[2, "a"]
[3, "a"]
[4, "a"]

6回

[1, "a"]
[2, "a"]
[3, "a"]
[4, "a"]
[0, "a"]
[1, "a"]
[2, "a"]
[3, "a"]

4(n-2)回

# n=1

[0, [:cat, [:rep, "a"], "a", "a", "a", "a"]]
[0, [:rep, "a"]]
[0, "a"]
[1, [:rep, "a"]]
[1, "a"]

a* が伸びていく

[1, "a"]
[0, "a"]
[1, "a"]

3回

1+2(n+1)=5回

計8回

# n=0

[0, [:cat, [:rep, "a"], "a", "a", "a", "a"]]

[0, [:rep, "a"]]

[0, "a"]

a* が伸びていく

[0, "a"]

1回

1+2(n+1)=3回

計4回

# 結果

tryが呼び出される回数は

- n = 0 のとき 4回
- n = 1 のとき 8回
- それ以上のとき 6n+1 回

# ざっと眺めた結果

- だいたいのひとは 6n+1 という式にたどりついている
- nが0,1のときを 4(n+1) という式にしたひとがけっこういる
  値としては合っているが、式に意味がない

- tryの呼出しを減らすためにtry_cat を変更したひとがいた
  問題の前提条件を変えるのは却下

# /a*aaaa...aaaa/

```
0.upto(20) {|n|
  m = count_try([:cat, [:rep, "a"]] + ["a"]*15,
                "a" * n)
  puts "#{n} #{m}"
}
```

```
0 4      5 34    10 89   15 168  20 253
1 8      6 43    11 103  16 185
2 13     7 53    12 118  17 202
3 19     8 64    13 134  18 219
4 26     9 76    14 151  19 236
```

# /a*aaa...aaa/でのグラフ