#12: Constraint Solving

Sankha Narayan Guria

EECS 700: Introduction to Program Synthesis

Why do we care?

1. Synthesis is combinatorial search, and so is SAT/SMT
2. SAT/SMT solvers are really good these days
3. ??? ← this week
4. Profit!!!

Boolean SATisfiability

gin ∨ tonic

Solution:

minor ↦ T
gin ↦ F
tonic ↦ T

Satisfiability Modulo Theories

$$(\text{gin} \lor \text{tonic}) \land (\text{age} < 21 \Rightarrow \text{abv} = 0) \land (\text{age} = 20)$$

In the United States, "gin" is defined as an alcoholic beverage of no less than 40% ABV...

Wikipedia

Satisfiability Modulo Theories

$$(\mathit{gin} \lor \mathit{tonic}) \land (\mathit{age} < 21 \Rightarrow \mathit{abv} = 0) \land (\mathit{age} = 20) \land (\mathit{gin} \Rightarrow \mathit{abv} \geq 40)$$

theory of Linear Integer Arithmetic

$$\begin{aligned} \mathit{age} &\mapsto 20 \\ \mathit{abv} &\mapsto 0 \\ \mathit{gin} &\mapsto F \\ \mathit{tonic} &\mapsto T \end{aligned}$$

Popular Solvers

Microsoft	Stanford	SRI	JKU Linz, Austria
Z3	CVC5	Yices2	Boolector

SMT competition: http://smtcomp.sourceforge.net

.smt2 // SMTLib format

```
(declare-fun age () Int)
(declare-fun abv () Int)
```

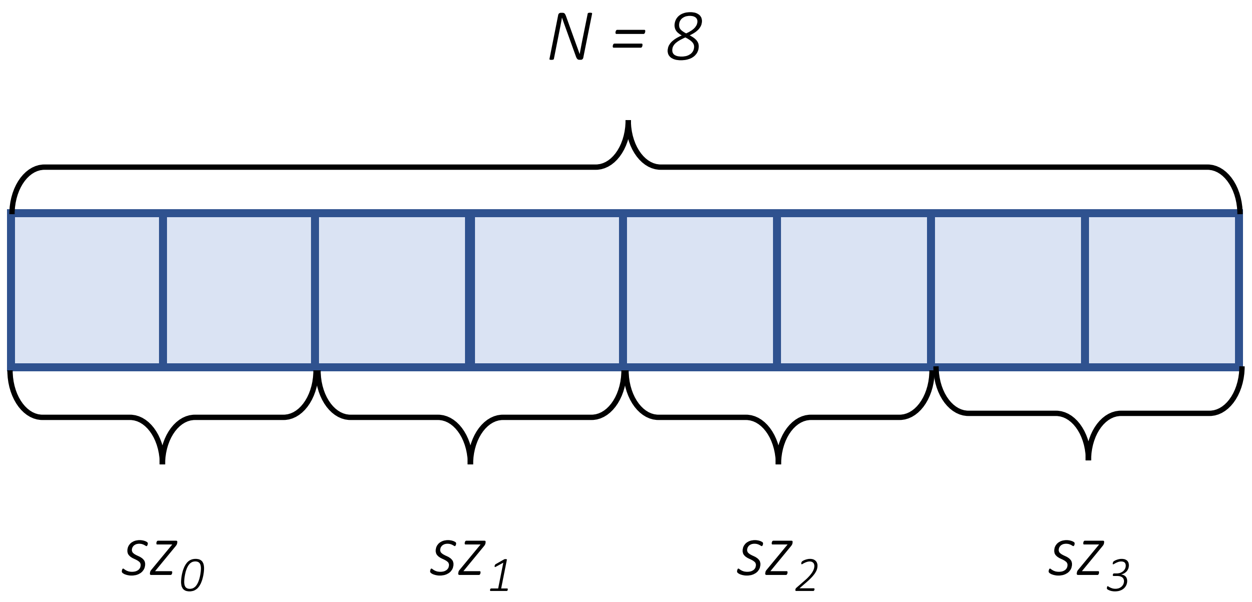
SMT-LIB

Uniform format for SMT problems understood by all solvers

```
(declare-fun age () Int)
(declare-fun abv () Int)
(declare-fun gin () Bool)
(declare-fun tonic () Bool)
(assert (or gin tonic))
(assert (implies (< age 21) (= abv 0)))
(assert (= age 20))
(assert (implies gin (>= abv 40)))
(check-sat)
(get-model)
```

Problem: Array Partitioning

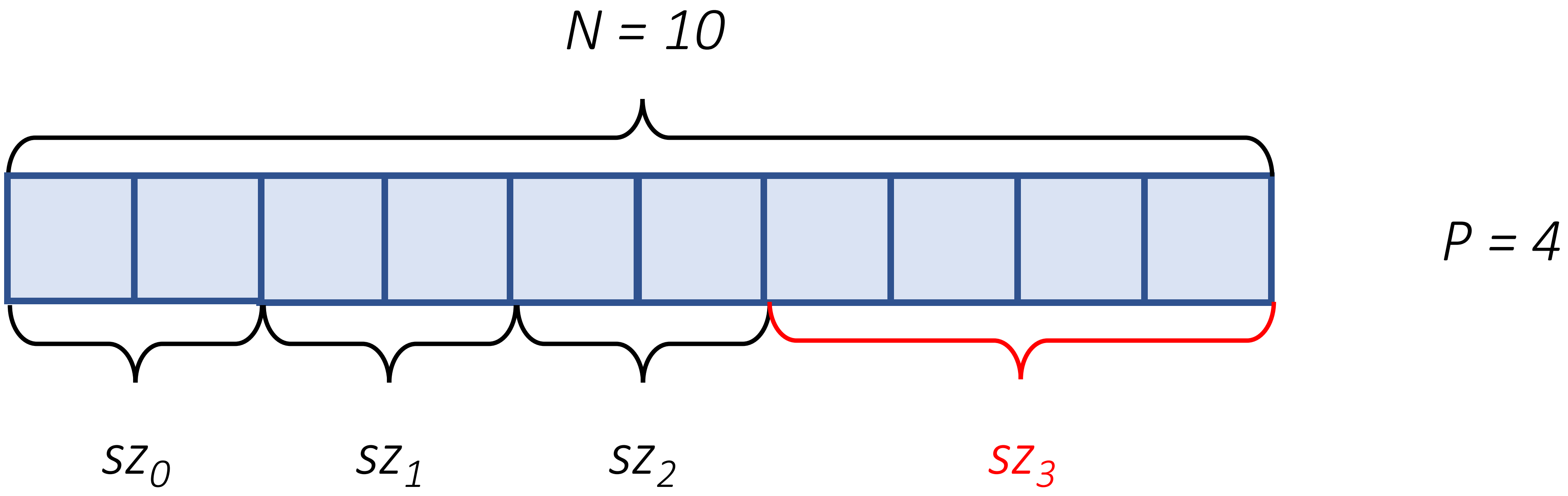
Partition an array of size *N* evenly into *P* sub-ranges



$P = 4$

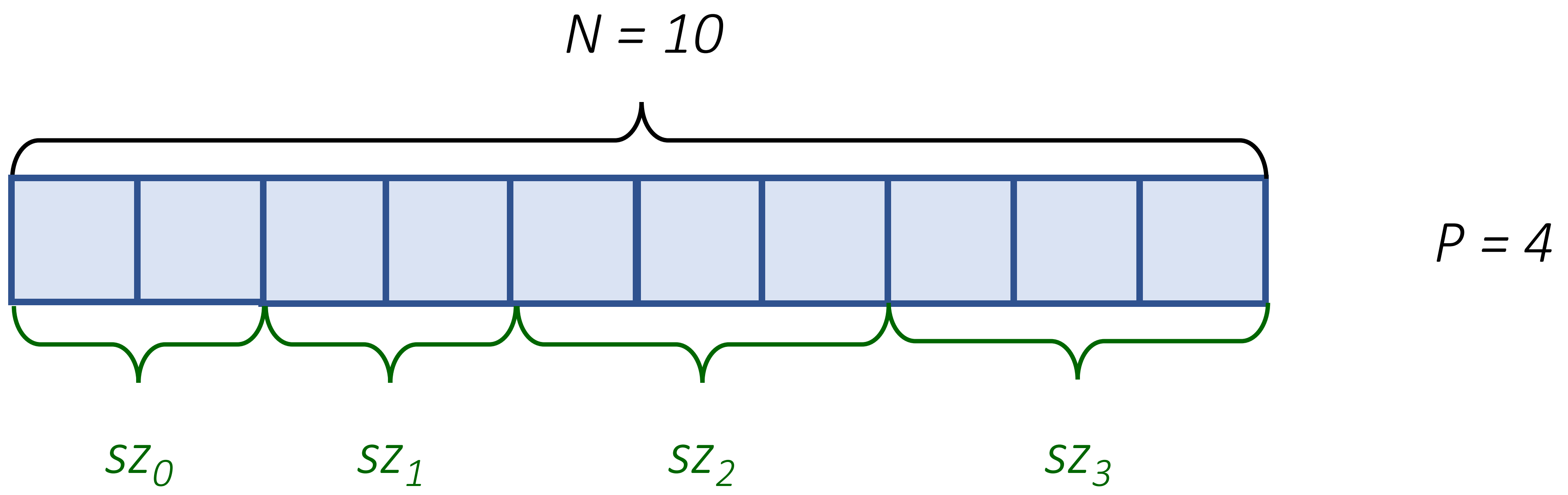
Problem: Array Partitioning

Partition an array of size N evenly into P sub-ranges



Problem: Array Partitioning

Partition an array of size N evenly into P sub-ranges



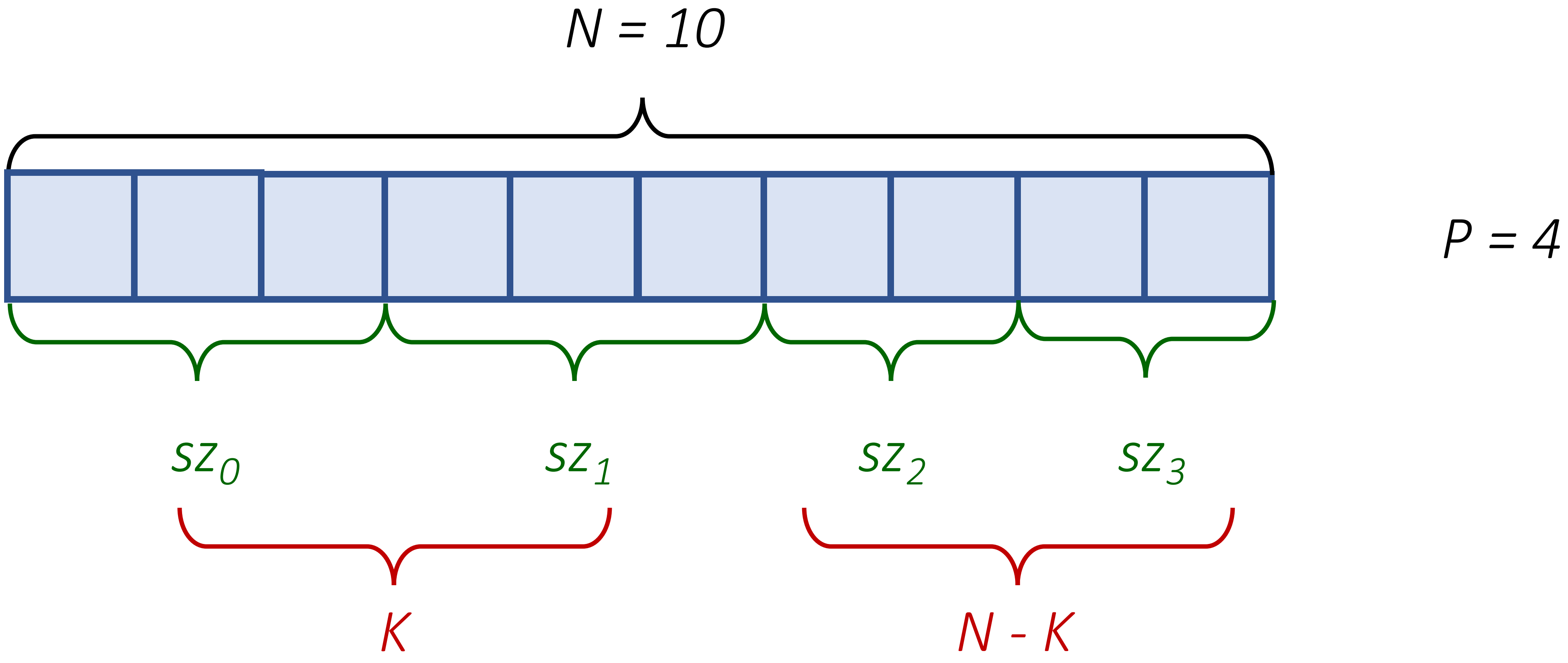
Can we always make them differ by at most 1?

Z3

to the rescue!

code: https://github.com/nadia-polikarpova/smt-talk

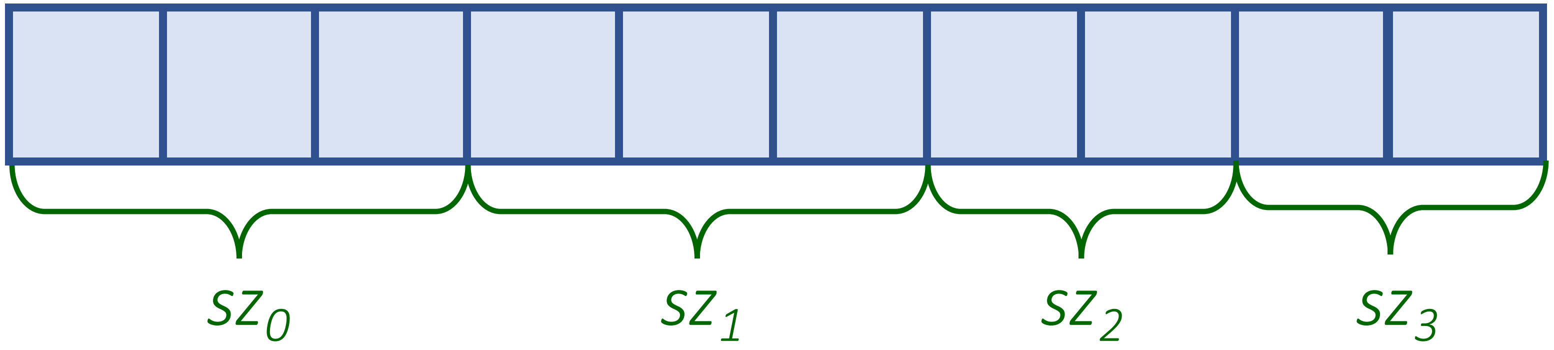
Let's generalize this into a program!



A program for partitioning

```
for i in range(P):
  if i < K:
    sz[i] = n/P + 1
  else:
    sz[i] = n/P
```

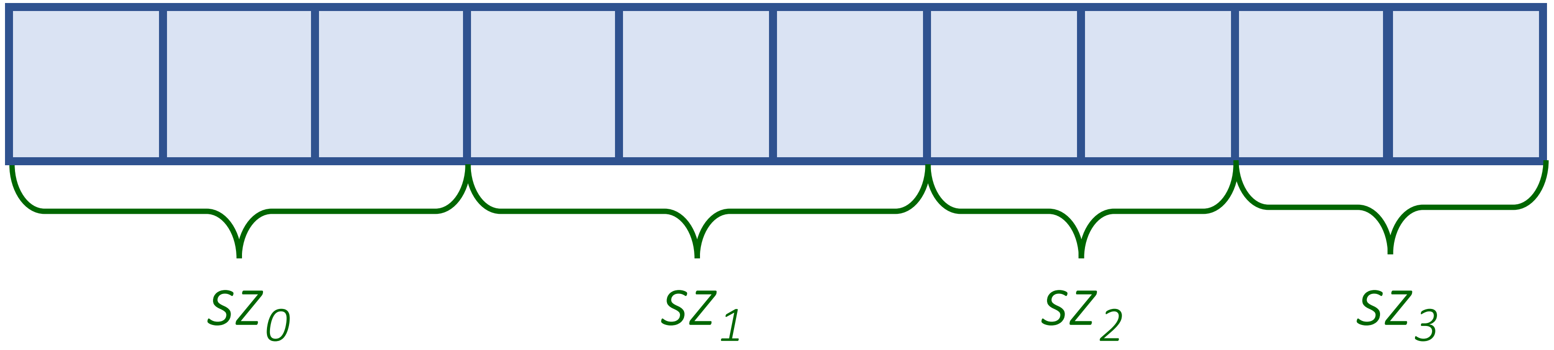
I want this program to work for all *n*!
What should my *K* be?



A program for partitioning

```
for i in range(P):
  if i < N % P:
    sz[i] = n/P + 1
  else:
    sz[i] = n/P
```

How do I prove that this program works for all *n*?



Verification with SMT

```
for i in range(P):
  if i < K:
    sz[i] = n/P + 1
  else:
    sz[i] = n/P
```

want: prove $\forall n.\, spec(n)$ $\quad$ $\forall n.\, \sum sz = n$

have: solve $\exists n.\, prop\,(n)$

idea: solve for counter-examples! $\quad$ $\exists n.\, \neg spec(n)$

$\exists n.\, \sum sz \neq n$