Selecting a Set

Peter Stuckey

Choosing from a set of objects

- Many problems require us to select a subset from a set of objects that
  - Meets some criteria; and
  - Optimizes some objective function
- Example 0-1 knapsack (at most one copy of each object)
  - Limit choices of x variable
  - Make the x an array of Booleans
  - Use a set variable

0-1Knapsack Model

```
int: n; % number of objects
set of int: OBJ = 1..n;
int: capacity;
array[OBJ] of int: profit;
array[OBJ] of int: size;

array[OBJ] of var 0..1: x;

constraint forall(i in OBJ)(x[i] >= 0);
constraint sum(i in OBJ)(size[i] * x[i])
           <= capacity;
solve maximize sum(i in OBJ)(profit[i] * x[i]);


output ["x = ", show(x), "\n"];
```

knapsack01.mzn

0-1Knapsack Model

```
int: n; % number of objects
set of int: OBJ = 1..n;
int: capacity;
array[OBJ] of int: profit;
array[OBJ] of int: size;

array[OBJ] of var bool: x;


constraint sum(i in OBJ)(size[i] *
                bool2int(x[i])) <= capacity;
solve maximize sum(i in OBJ)
                  (profit[i] * bool2int(x[i]));

output [”x = “, show(x), “\n”];
```

knapsack01bool.mzn

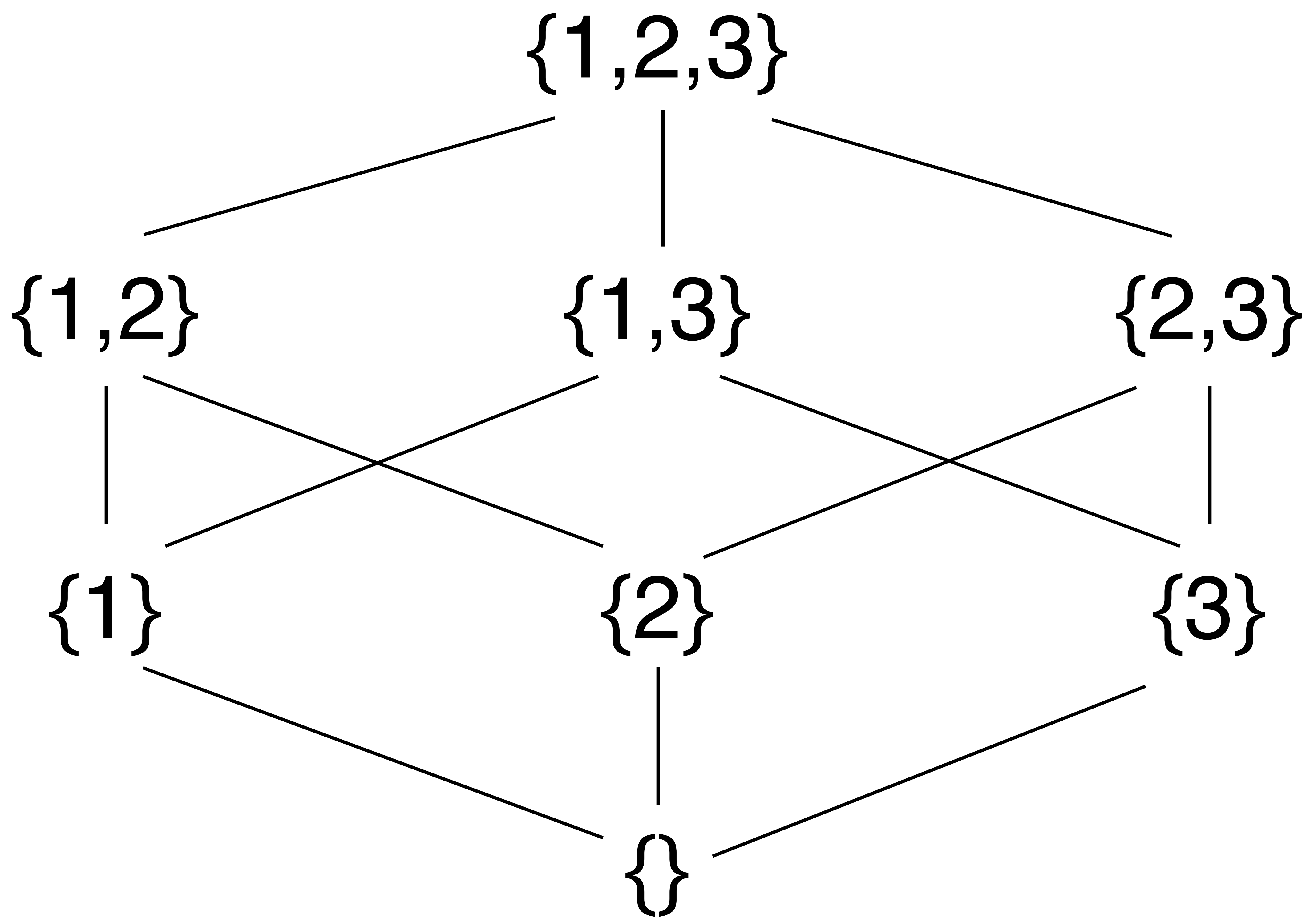
0-1Knapsack Model

```
int: n; % number of objects
set of int: OBJ = 1..n;
int: capacity;
array[OBJ] of int: profit;
array[OBJ] of int: size;

var set of OBJ: x;


constraint sum(i in x)(size[i]) <= capacity;

solve maximize sum(i in x)(profit[i]);


output [”x = “, show(x), “\n”];
```

knapsack01set_concise.mzn

Set Variables

- Set variables in MiniZinc choose a set from a given fixed superset
- `var set of {1,2,3}: x;`



Set Constraints

- MiniZinc provides the (infix) set operations
  - in, (membership e.g. x in s)
  - subset, superset
  - intersect (intersection)
  - union
  - card (cardinality)
  - diff (set difference e.g x diff y = x \ y)
  - symdiff (symmetic difference)
    - e.g. {1, 2, 5, 6} symdiff {2, 3, 4, 5} = {1, 3, 4, 6}

Which model is best?

- Most solvers will treat each model the same
  - CP solvers may treat the last model better since they can combine cardinality reasoning with other set reasoning
- Model whichever makes it easier to express the constraints
  - for knapsack01 the first version
- Model using the highest level model
  - the last version

Choosing a Set Representation: Example

Peter Stuckey

SetSelect Question

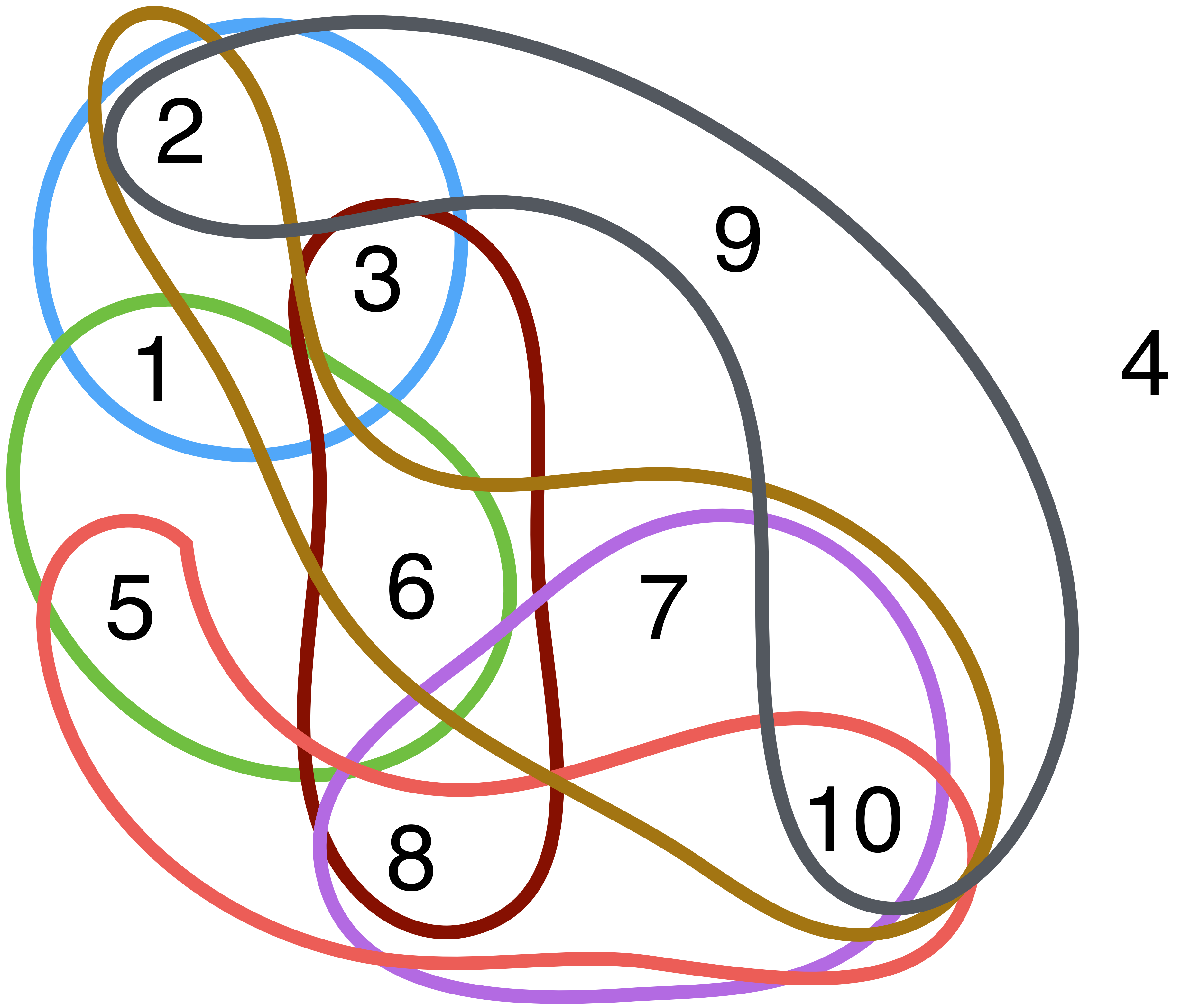
- Write a MiniZinc model that given an array of `k` subsets of numbers `1..n`, chooses a subset of `1..n` which includes at most one from each subset and maximizes the sum of the chosen set.

```
n = 10;
k = 7;
s = [{1,5,6}, {2,6,7,10}, {3,6,8}, {1,2,3},
     {2,9,10}, {5,8,10}, {7,8,10}];
```

setselect.dzn

Simple Set Select Algorithm

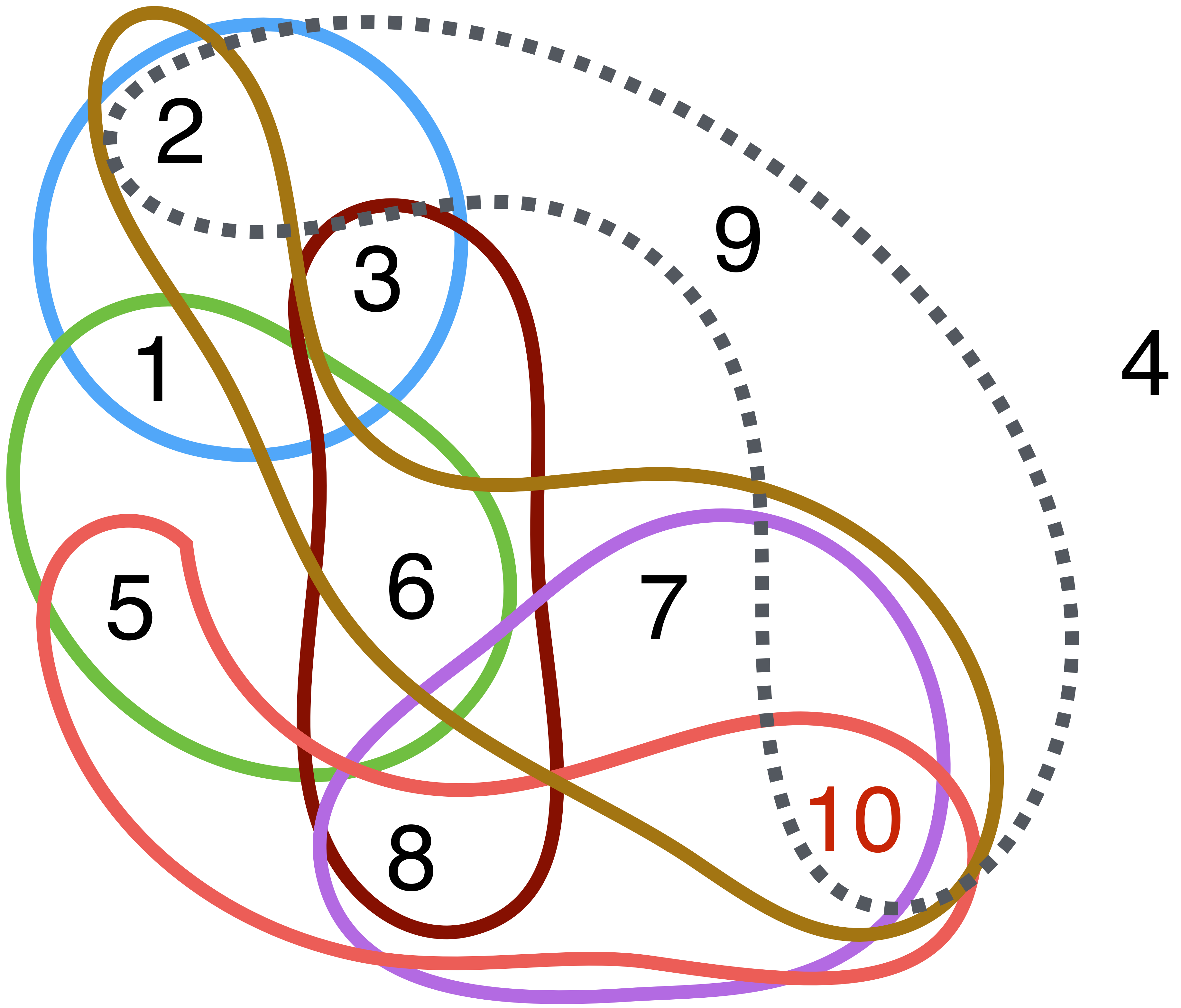
- Greedy algorithm
  - choose the largest available element
  - eliminate choices that are no longer valid




```
n = 10;
k = 7;
s = [{1,5,6}, {2,6,7,10}, {3,6,8}, {1,2,3},
     {2,9,10}, {5,8,10}, {7,8,10}];
```

Simple Set Select Algorithm

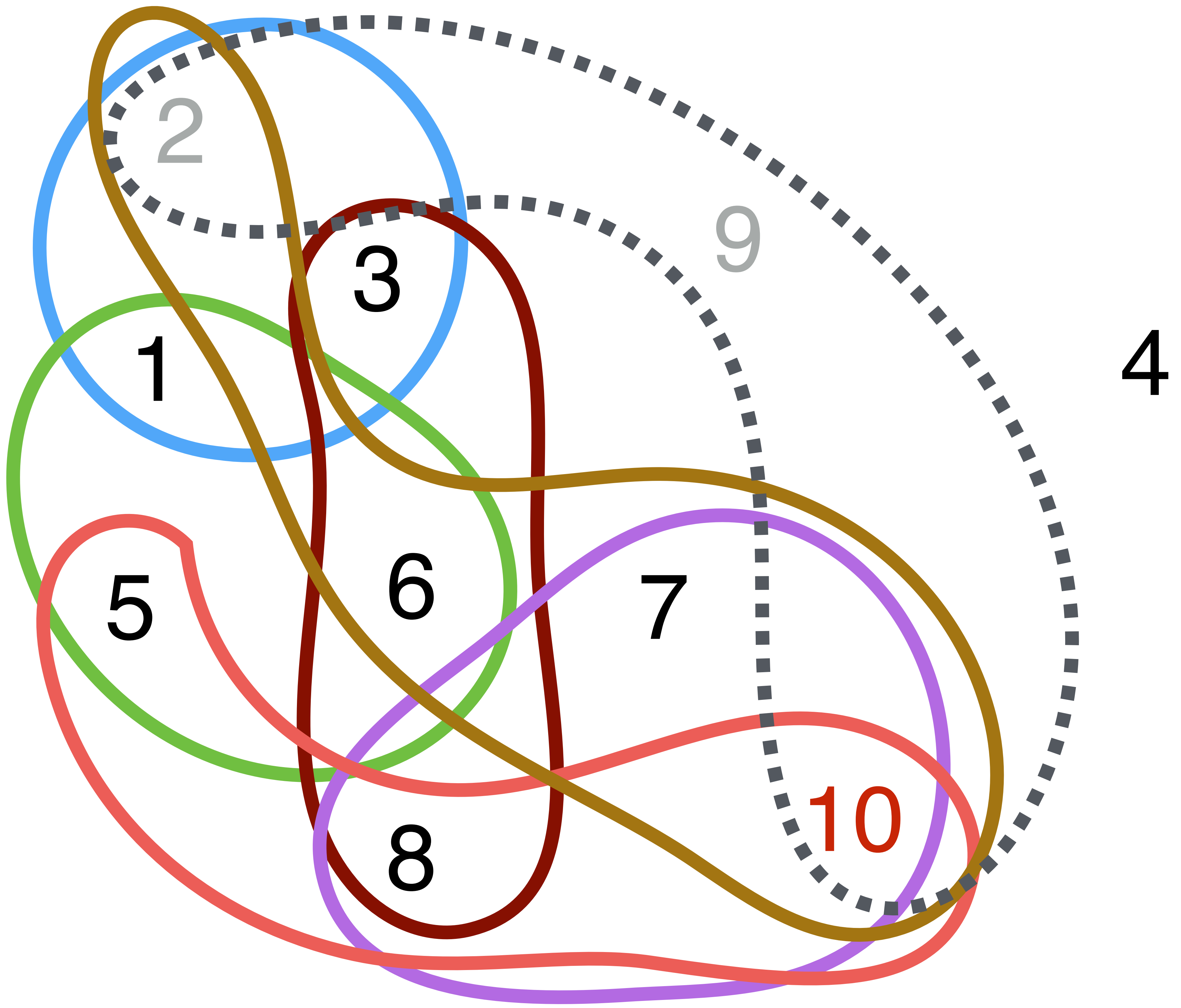
- Greedy algorithm
  - choose the largest available element
  - eliminate choices that are no longer valid




```
n = 10;
k = 7;
s = [{1,5,6}, {2,6,7,10}, {3,6,8}, {1,2,3},
      {2,9,10}, {5,8,10}, {7,8,10}];
```

Simple Set Select Algorithm

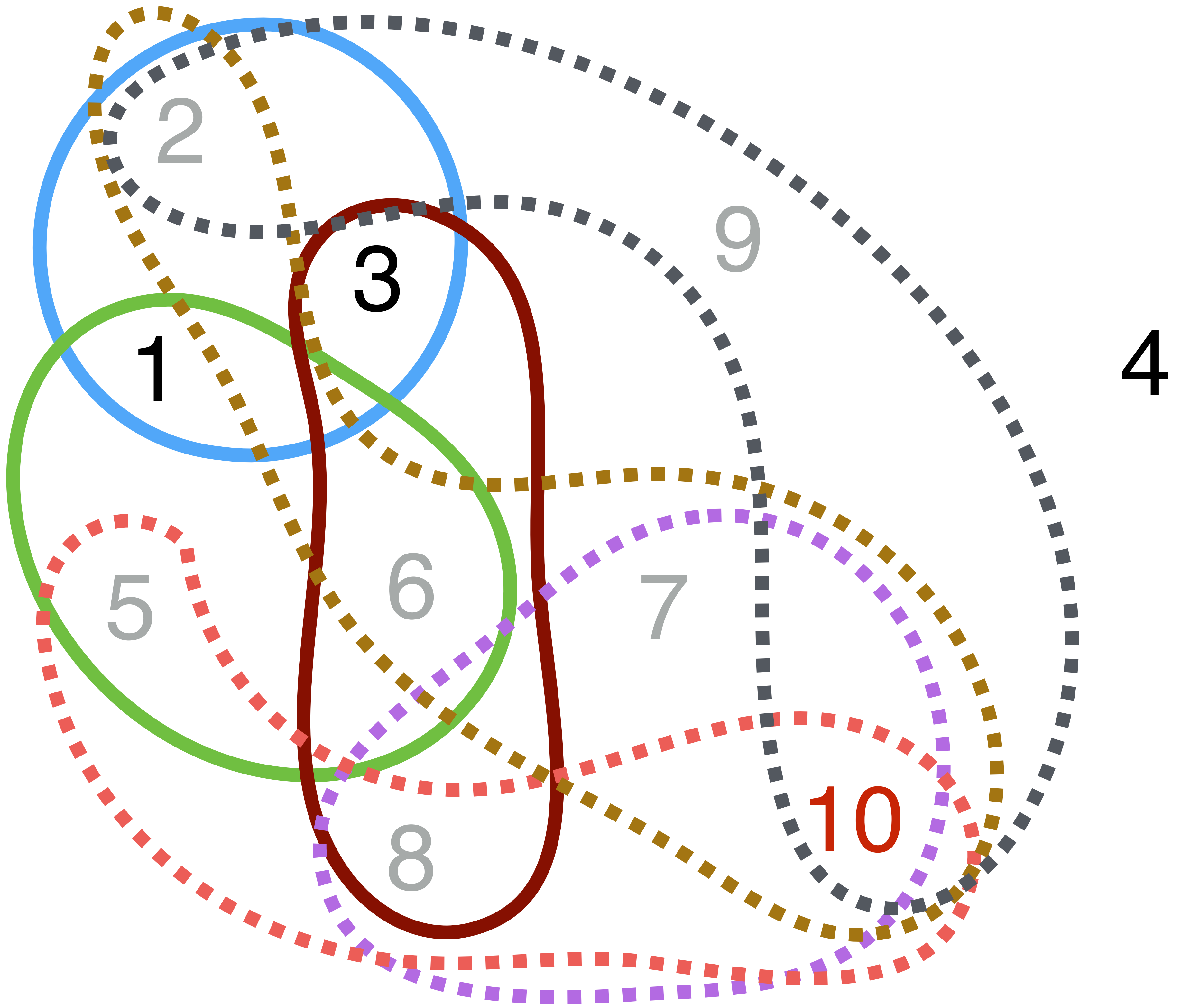
- Greedy algorithm
  - choose the largest available element
  - eliminate choices that are no longer valid




```
n = 10;
k = 7;
s = [{1,5,6}, {2,6,7,10}, {3,6,8}, {1,2,3},
     {2,9,10}, {5,8,10}, {7,8,10}];
```

Simple Set Select Algorithm

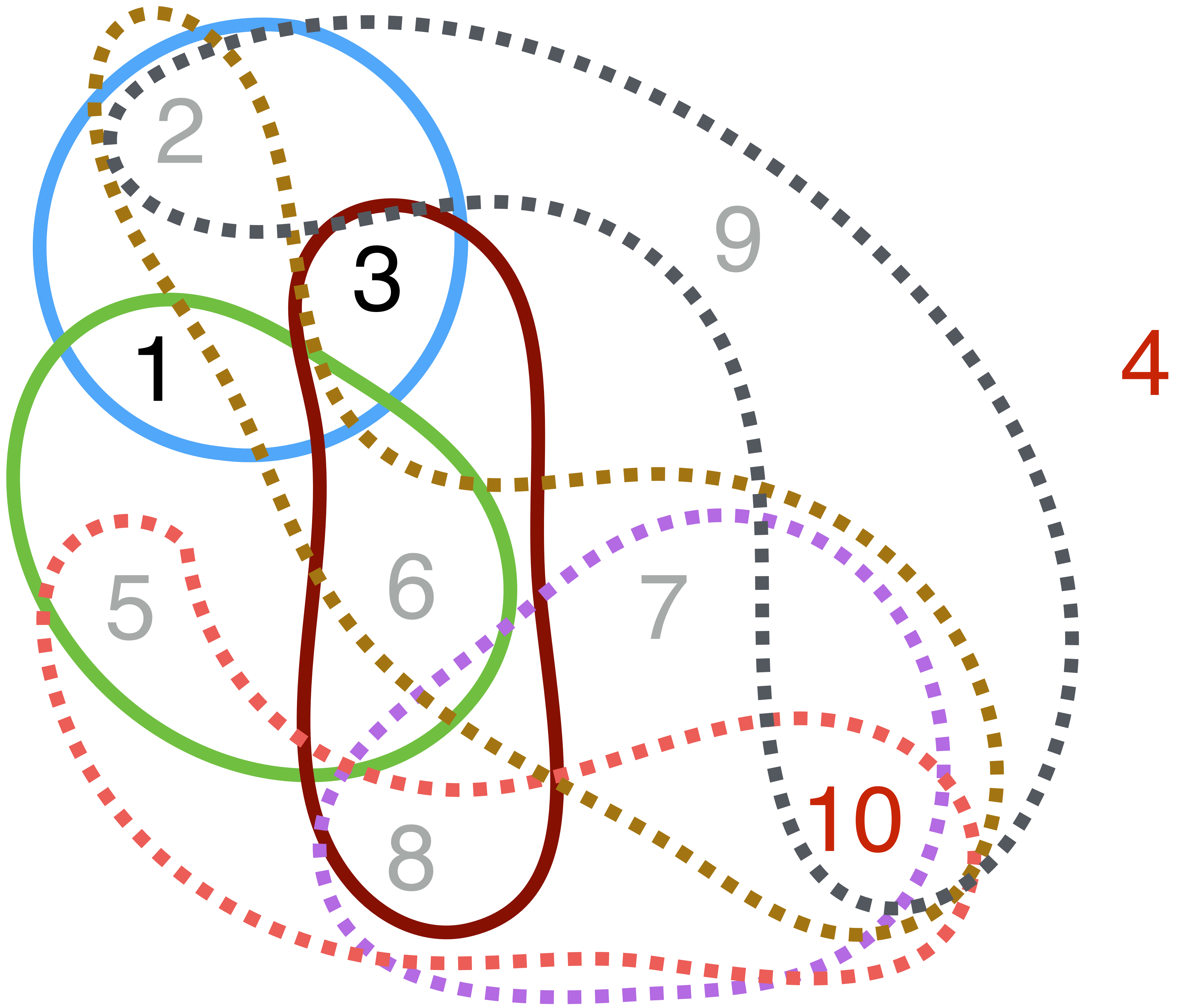
- Greedy algorithm
  - choose the largest available element
  - eliminate choices that are no longer valid




```
n = 10;
k = 7;
s = [{1,5,6}, {2,6,7,10}, {3,6,8}, {1,2,3},
     {2,9,10}, {5,8,10}, {7,8,10}];
```

Simple Set Select Algorithm

- Greedy algorithm
  - choose the largest available element
  - eliminate choices that are no longer valid

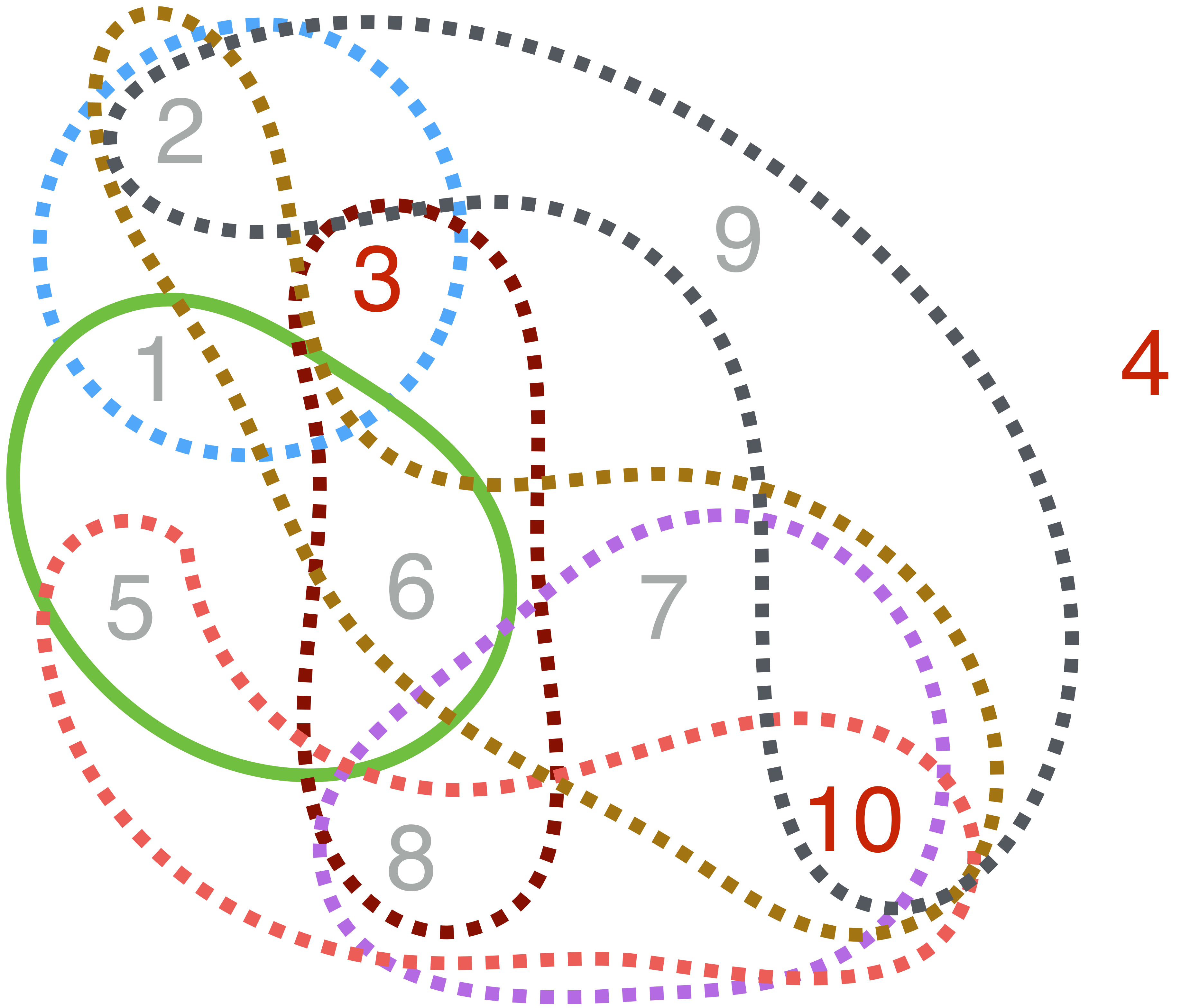



```
n = 10;
k = 7;
s = [{1,5,6}, {2,6,7,10}, {3,6,8}, {1,2,3},
     {2,9,10}, {5,8,10}, {7,8,10}];
```

Simple Set Select Algorithm

- Greedy algorithm
  - choose the largest available element
  - eliminate choices that are no longer valid

{3,4,10}




```
n = 10;
k = 7;
s = [{1,5,6}, {2,6,7,10}, {3,6,8}, {1,2,3},
     {2,9,10}, {5,8,10}, {7,8,10}];
```

SetSelect Data + Decisions

- Data

```
int; n;
set of int: OBJ = 1..n;
int: k;
set of int: SET = 1..k;
array[SET] of set of OBJ: s;
```

- Decisions

```
var set of OBJ: x;
```

SetSelect Constraints + Objective

- At most one intersection

```
forall(i in SET)
         (card(x intersect s[i]) <= 1);
```

- Objective

```
solve maximize sum(i in x)(i);
```

Solving the model

- Executing the model

```
$ minizinc setselect.mzn setselect.dzn

x = {3,4,5,7,9};
----------
==========
```

- Much better than the greedy algorithm

SetSelect Revised Question

- Write a MiniZinc model that given an array of `k` subsets of numbers `1..n`, chooses a subset of `1..n` of size `u` which includes at most one from each subset and maximizes the sum of the chosen set.

```
n = 10;
k = 7;
u = 3;
s = [{1,5,6}, {2,6,7,10}, {3,6,8}, {1,2,3},
     {2,9,10}, {5,8,10}, {7,8,10}];
```

setselectr.dzn

SetSelect Revised Addition

- Additional constraint: `card(x) = u;`

- Executing the model

```
$ minizinc setselectr.mzn setselectr.dzn

x = {4,8,9};
----------
==========
% failures = 237
```

- But we can model a set of known cardinality differently!

Choosing a Fixed Cardinality Set

Peter Stuckey

SetSelect Revised Question

- Write a MiniZinc model that given an array of `k` subsets of numbers `1..n`, chooses a subset of `1..n` of size `u` which includes at most one from each subset and maximizes the sum of the chosen set.

```
n = 10;
k = 7;
u = 3;
s = [{1,5,6}, {2,6,7,10}, {3,6,8}, {1,2,3},
     {2,9,10}, {5,8,10}, {7,8,10}];
```

setselectrev.dzn

Deciding a set of fixed cardinality

- Instead of a set variable `var set of 1..n: x` with
  - cardinality constraint `card(x) = u`
- An array of `u` values

  `array[1..u] of var 1..n: x`
  - and some other constraints …
- Why: suppose n = 1000, u = 4
- First representation
  - 1000 Boolean variables
- Second representation
  - 4 integer variables

SetSelect Revised Question

- Write a MiniZinc model that given an array of `k` subsets of numbers `1..n`, chooses a subset of `1..n` of size `u` which includes at most one from each subset and maximizes the sum of the chosen set.

```
n = 10;
k = 7;
u = 3;
s = [{1,5,6}, {2,6,7,10}, {3,6,8}, {1,2,3},
     {2,9,10}, {5,8,10}, {7,8,10}];
```

setselectrev.dzn

SetSelect Revised Model

- Decisions

```
array[1..u] of var 1..n: x;
for(i in 1..u-1)(x[i] < x[i+1]);
```

- At most one intersection

```
forall(i in SET)
      (sum(j in 1..u)
           (x[j] in s[i]) % coercion
       <= 1);
```

- Objective

```
solve maximize sum(x);
```

Solving the model

- Executing the model

```
$ minizinc setselectr2.mzn setselectr.dzn

x = [5,7,9];
----------
==========
% failures = 22
```

- This representation makes search easier

Overview

- There are multiple ways to represent fixed cardinality sets
  - var set of OBJ + cardinality constraint
    - good if the solver natively supports sets
    - good when OBJ is not too big
  - array[1..u] of var OBJ
    - good when u is small
- Two critical issues in modelling decisions
  - ensure each solution to the model is a solution of the problem
  - try to ensure each solution of the problem only has one solution in the model (symmetry)

Choosing a Bounded Cardinality Set

Peter Stuckey

SetSelect Revised Question

- Write a MiniZinc model that given an array of `k` subsets of numbers `1..n`, chooses a subset of `1..n` of size at most `u` which includes at most one from each subset and maximizes the sum of the chosen set.

```
n = 10;
k = 7;
u = 3;
s = [{1,5,6}, {2,6,7,10}, {3,6,8}, {1,2,3},
     {2,9,10}, {5,8,10}, {7,8,10}];
```

setselectrev.dzn

Deciding a set of bounded cardinality

- Instead of a set variable `var set of OBJ: x` with
  - cardinality constraint `card(x) <= u`
- An array of `u` values

  `array[1..u] of var EOBJ: x`

  - extended OBJ: EOBJ = OBJ ∪ { extra-value }
  - extra value represents: no element
- For example: OBJ = 1..n
  - EOBJ = 0..n

Two critical issues

- Each solution in the model represents a solution in the problem
  - [3,0,3] ❌ no repeated values
  - [0,2,0] ✅ repeated extra values are OK

- Each solution in the problem has just one solution representative in the model
  - [0,2,0], [0,0,2], [2,0,0] = {2} ❌
  - [0,1,2], [0,2,1], [1,0,2], [1,2,0], [2,0,1], [2,1,0] ❌
- Add constraints to fix these issues

SetSelect Revised Question

- Write a MiniZinc model that given an array of `k` subsets of numbers `1..n`, chooses a subset of `1..n` of size at most `u` which includes at most one from each subset and maximizes the sum of the chosen set.

```
n = 10;
k = 7;
u = 3;
s = [{1,5,6}, {2,6,7,10}, {3,6,8}, {1,2,3},
     {2,9,10}, {5,8,10}, {7,8,10}];
```

setselectrev.dzn

SetSelect Revised Question

- Decisions

```
array[1..u] of var 0..n: x;
for(i in 1..u-1)
   (x[i] >= bool2int(x[i]=0)+ x[i+1]);
```

- At most one intersection

```
forall(i in SET)
      (sum(j in 1..u)
          (x[j] in s[i]) % 0 is safe
       <= 1);
```

- Objective

```
solve maximize sum(x); % 0 is safe
```