情報知識ネットワーク特論/IKN Special Lecture Data Mining 2b: Subset enumeration データマイニング2b:補足:部分集合の列挙

有村 博紀, 喜田拓也
北海道大学大学院 情報科学研究科 コンピュータサイエンス専攻
email: {arim,kida}@ist.hokudai.ac.jp
http://www-ikn.ist.hokudai.ac.jp/~arim

Slide: http://www-ikn.ist.hokudai.ac.jp/ikn-tokuron/

今日の内容

2回(補足): 部分集合の列挙

- 前回の演習問題
- 集合の基本
- 部分集合列挙
- アルゴリズム
  - 整数との1対1対応
  - 二分割法(分割統治法)
  - 逆探索法(バックトラック法)

前回の演習

- 練習1．集合をS = {1, 2, 3, 4}とする．次の問題に答えよ．
  Let S = {1, 2, 3, 4} be a finite set of four elements.

1. 空集合φはSの部分集合か？　Is the emptyset φ a subset of S?
2. Sのすべての部分集合を書き出せ. Write/List all subsets of S.
3. それらの総数はいくつか？
   How many are they?
4. オプション問題）非負整数 n を入力としてもらい，集合S = {1, ..., n }のすべての部分集合を書きだすプログラムを書け．プログラミング言語は何を使ってもよい．日本語や英語の疑似コードでもかまわない．
   Optional Problem) Write a computer program (algorithm) that receives a nonnegative number *n*, and prints all subsets of the set S = {1, ..., *n*}. You can use any programing languages as well as pseudo code written in Japanese/English.

集合の基本

- 集合 (set)とはモノの集まりAで，任意の要素xと集合Aに対して，所属関係"$x \in A$"がYESかNOに決定する
- 空集合Φ: 任意のxに対して"$x \notin \Phi$"となる集合
- 例：
  - $\Phi = \{\}$ （空集合）
  - $A = \{1, 2, 3\}$ （外延記法）
  - $E = \{ x \mid x$を2で割った余りが0 $\}$ （内包記法）

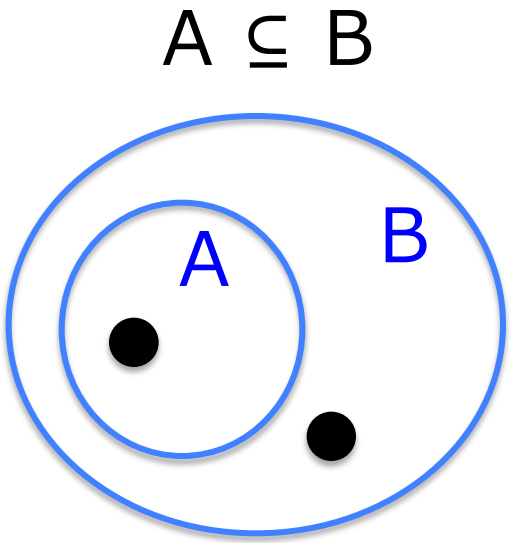
集合の基本

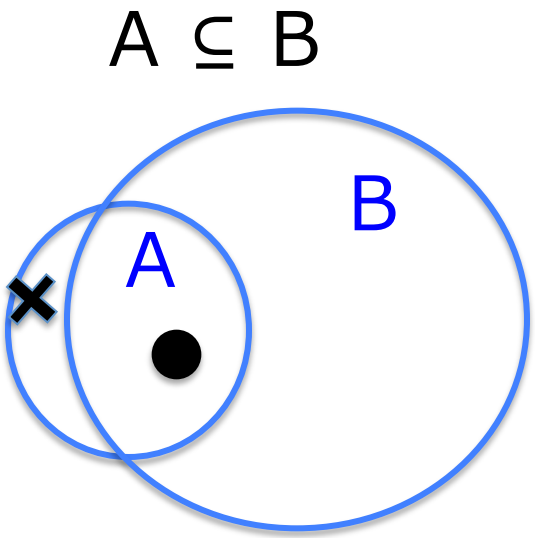
- 定義：AがBの部分集合である：

  $A \subseteq B$

  $\Leftrightarrow \forall x\,[x \in A \Rightarrow x \in B\,]$

  ⇔「すべての要素xに対して，xが集合Aに入るならば集合Bにも入る」





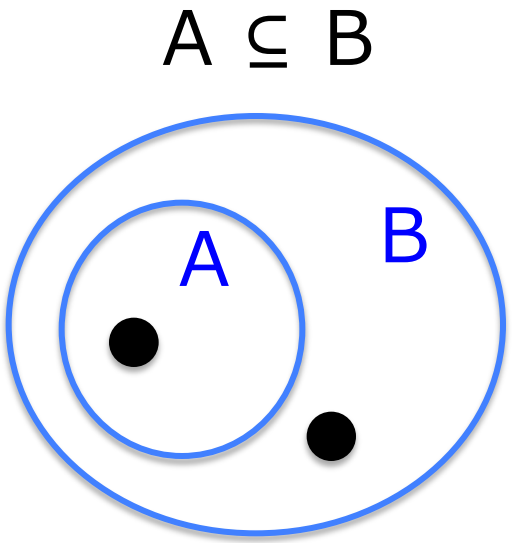
集合の基本

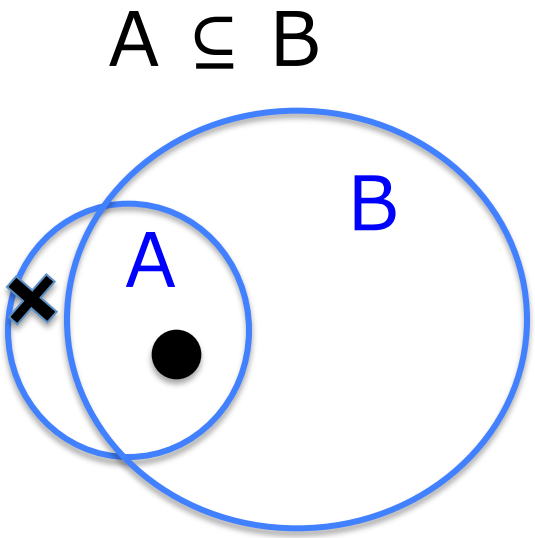
- 定義：φがBの部分集合である：

  φ ⊆ B

  ⇔ ∀x [x∈φ ⇒ x∈B]

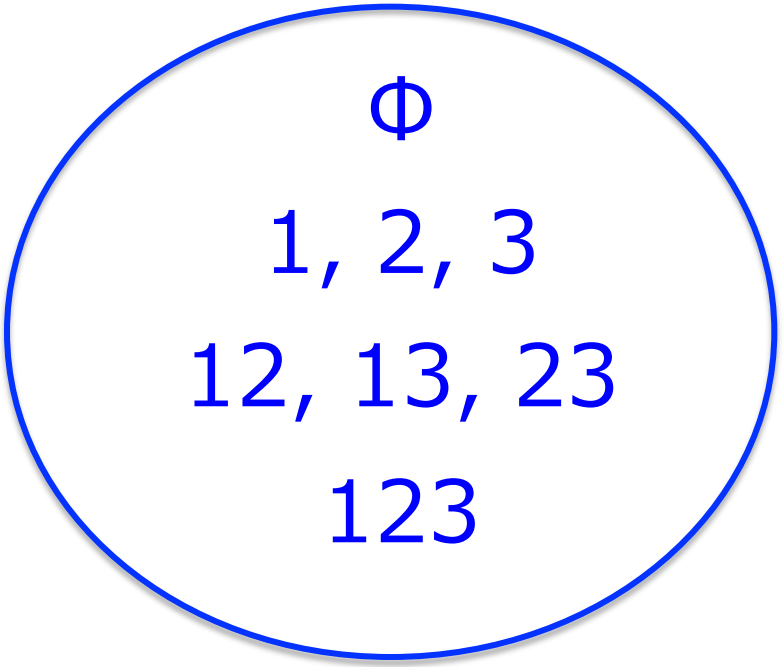
  ⇔「すべての要素xに対して，xが空集合φに入るならば集合Bにも入る」





集合Sのべき集合

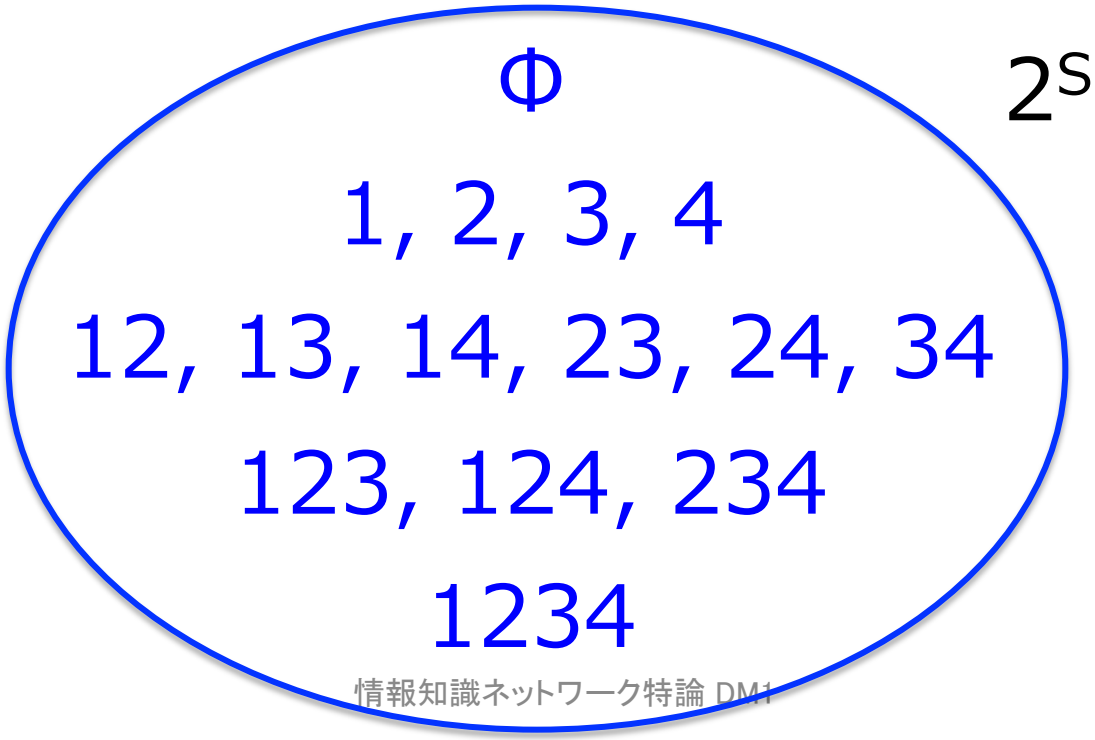
- Sのべき集合(power set)
  $2^S = \{ X \mid X \subseteq S \}$＝Sの全ての部分集合の族
- 例：S = {1, 2, 3}



2^S

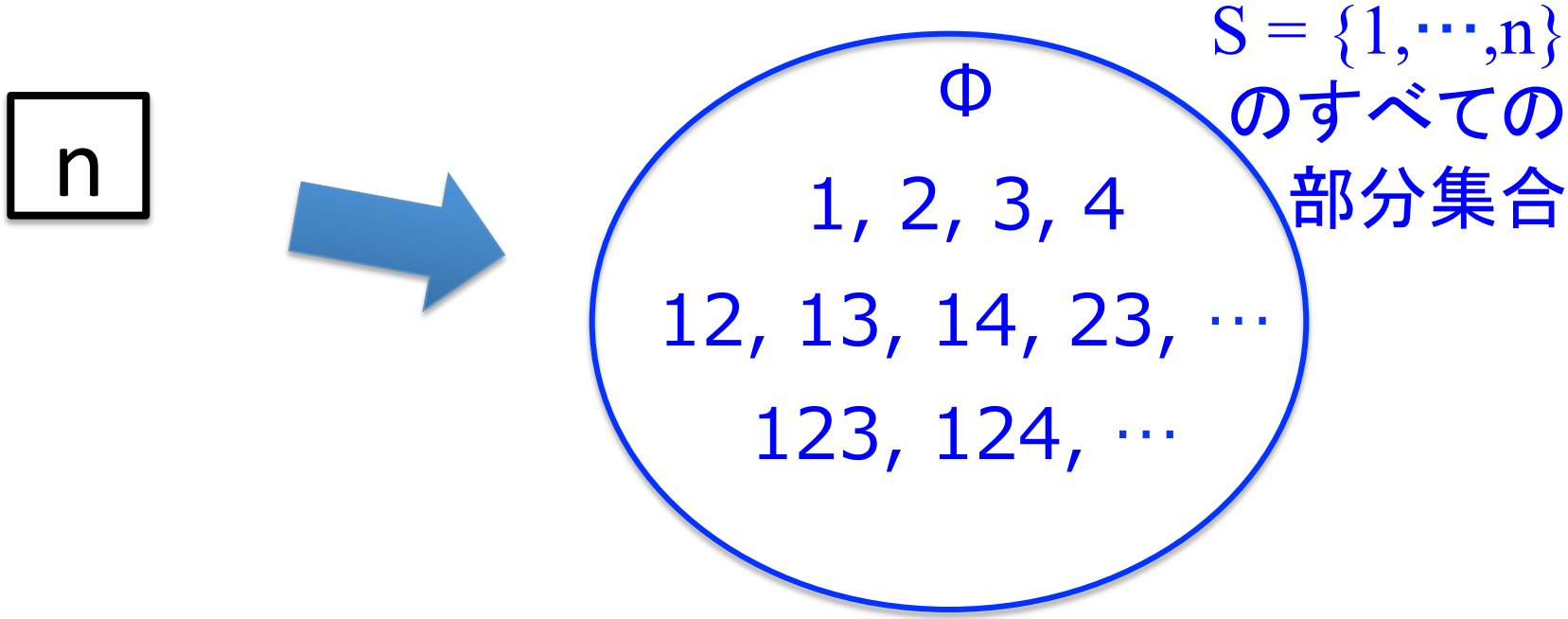
集合Sの部分集合

- Sのべき集合(power set)
  $2^S = \{ X \mid X \subseteq S \}$＝Sの全ての部分集合の族
- 例：S = {1, 2, 3, 4}



議論：演習の問2

入力nを受け取り、S = {1,…,n}のすべての部分集合を書き出せ.
List all subsets of S.



部分集合列挙1：数との１対1対応

数	ビットベクトル（123の順）	部分集合
0	000	Φ
1	001	3
2	010	2
3	011	2,3
4	100	1
5	101	1,3
6	110	1,2
7	111	1,2,3

- S = {1,…,n}の部分集合全体は，0から2^n-1の整数全体と対応する
- 性質：整数iがSに含まれる(含まれない) ⇔ ビットiが1 (が0)

1．1対1対応法のプログラム

```
#subset2.rb
#一対一法でS={1,...,n}の部分集合を出力する
n = ARGV[0].to_i
print "input: n=#{n}¥n"

def enum(x, n)
  k = (2**n) -1
  for m in 0..k do
    x = ntob(m, n)
    print "#{x}¥n"
  end
end

#メイン文：入力nは集合サイズ
enum("", n)

#EOF
```

```
### 実行例

$ ruby subset1.rb 3

input: n=3
order of elements: 321
000
001
010
011
100
101
110
111
$
```

議論：

基本問題：

Sのすべての部分集合を書き出せ.

List all subsets of S.

現実には、いろいろな応用がある

- 要素数k以下のすべての部分集合（例：k=3）
- 与えられた要素iを含むすべての部分集合
- ある制約を満たすすべての部分集合
- （例：制約＝データベース中での「頻度」が指定値以上）

議論：

基本問題：

Sのすべての部分集合を書き出せ.
List all subsets of S.

一対一法だけで十分か？

部分集合列挙2：二分割法

ビットベクトル（123の順）	部分集合
000	Φ
001	3
010	2
011	2,3
100	1
101	1,3
110	1,2
111	1,2,3

- S の部分集合全体が答え全体（解集合）
- 答え全体を二つに分けて，分割統治法で列挙する
- 各要素 i∈Sに対し，答え全体を、iを含むものと含まないもの二つに分けて再帰的に列挙する

2. 二分割法のプログラム

```ruby
#subset2.rb
#二分割法でS={1,...,n}の部分集合を出力する
#再帰てつづき
def enum(x, n)
  if n == 0 then
    print "'#{x}'¥n""   #集合xを出力
  else
    a = n.to_s
    enum(x+a, n-1)
    #=> nを含む部分集合をすべて出力
    enum(x, n-1)
    #=> #nを含まない部分集合をすべて出力
  end
end

#メイン文：入力nは集合サイズ
enum("", n)

#EOF
```

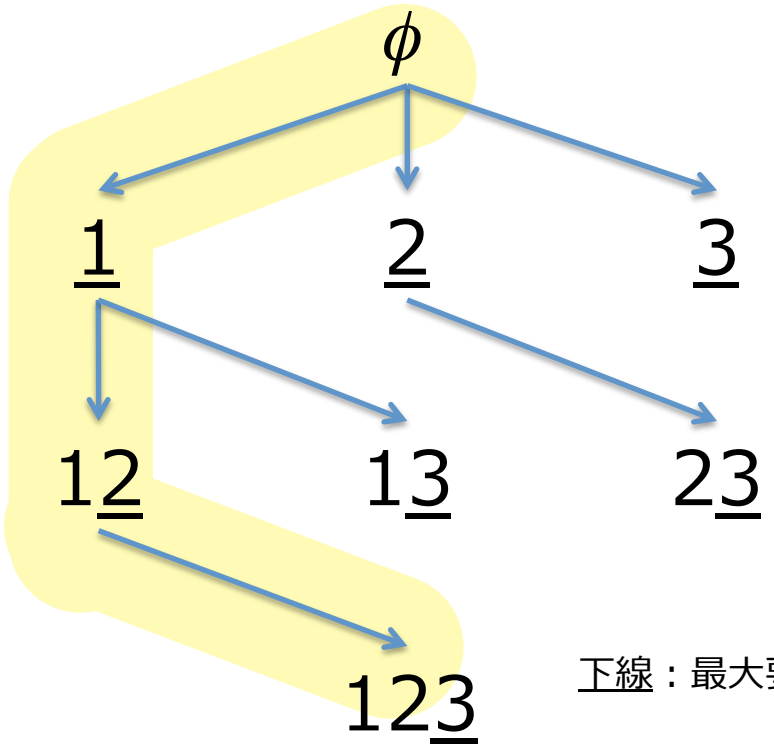
```
### 実行例

$ ruby subset2.rb 3

subset2.rb: printing all
subsets of S={1,..,3}
'321'
'32'
'31'
'3'
'21'
'2'
'1'
''
```

部分集合列挙3：逆探索法

基本アイディア：部分集合がなす空間上に，木の形をした探索路をつくり，その上を探索する.



下線：最大要素

概要：

- 空でない全ての部分集合Xに対して，その一意な「親」P(X)を一つ定める.
  - P(X) := X – {max(X)}
- 親ポインタ全体は，根φをもつ全部分集合上の根付き木を定める.
- アルゴリズム：この木を，根からスタートして．再帰でバックトラックしながら探索する.

探索では，現在の集合Xに，その最大要素 k = max(X) より大きな要素 i > kを一つ加えたものを「子」Y := X U {i} として再帰的に空間を探索する

3 逆探索法のプログラム

```
#subset3.rb
#逆探索法でS={1,...,n}の全ての部分集合を出力する
#再帰てつづき
def enum(x, k, n)
  print "'#{x}'¥n"      #集合xを出力
  if k == n then
    return
  end
  for i in k+1..n do
    a = i.to_s          #文字に変換
    enum(x+a, i, n)
  end
end

#メイン文：入力nは集合サイズ
enum("", 0, n)

#EOF
```

```
### 実行例

$ ruby subset3.rb 3

subset3.rb: printing all subsets of S={1,..,3}
''
'1'
'12'
'123'
'13'
'2'
'23'
'3'
```

Python 2.7.9. Mac OS X, 0.10.5

補足：部分集合列挙：共通ヘッダー

```
# coding: utf-8
# S = {1,...,n}の全ての部分集合を出力する

#入力nの読み込み
if ARGV.size() < 1 then
  print "usage: #{$0} n¥n"
  exit 1;
end
n = ARGV[0].to_i
print "#{$0}: printing all subsets of S={1,..,#{n}}¥n”

#このあとにそれぞれの手続き定義
```

- 上記のプログラムを実行するための共通ヘッダー
- 入力nのコマンドラインからの読み込みを行う.
- 各プログラムの先頭に，左のソースコードを貼り付けて，ソースコードとして，シェル上でrubyコマンドで実行する.

Ruby言語の表記:
- 行頭が‘＃’文字の行はコメント
- ARGV => コマンド引数 ARGV[0], ARGV[1], ...の配列
- var.to_i =>整数に変換.
- var.to_s => 文字列に変換
- #{var} => 変数varの値を文字列中に展開

まとめ

2回(補足): 部分集合の列挙

- 集合の基本
- 部分集合列挙
- アルゴリズム
  - 整数との1対1対応
  - 二分割法(分割統治法)
  - 逆探索法(バックトラック法)