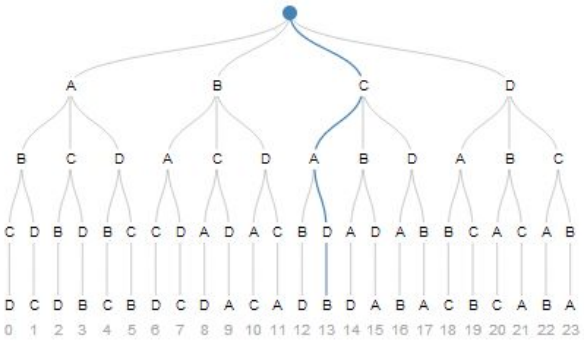
121 lines (90 sloc) | 3.64 KB

EOPL Midterm

Alireza Habibzadeh 99109393

2

i



از درخت تصمیم استفاده می‌کنیم.

مطابق شکل در هر مرحله از بین اعضای باقی‌مانده یکی را انتخاب می‌کنیم و سپس آن عضو را از انتخاب‌هایمان حذف می‌کنیم. آن قدر این کار را ادامه می‌دهیم تا لیست انتخاب‌های ممکنمان تهی شود. هر راس می‌تواند بیش از دو فرزند داشته باشد پس این درخت چندتایی است و دودویی نیست.

```
#lang racket
(require (lib "eopl.ss" "eopl"))

(define-datatype Permtree Permtree?
  [root (children (list-of Permtree?))]
  [node (num integer?) (children (list-of Permtree?))]
  [leaf (num integer?)])
```

ii

```
#lang racket

(define (permtree-helper l n)
  (if (null? l)
      (leaf n)
      (node n (map (λ (x) (permtree-helper (remove x l) x)) l))))

(define (ListToPermTree L)
  (root (map (λ (x) (permtree-helper (remove x L) x)) L)))
```

```
Welcome to DrRacket, version 8.5 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (ListToPermTree '(1 2 3))
(root
 (list
  (node 1 (list (node 2 (list (leaf 3))) (node 3 (list (leaf 2)))))
  (node 2 (list (node 1 (list (leaf 3))) (node 3 (list (leaf 1)))))
  (node 3 (list (node 1 (list (leaf 2))) (node 2 (list (leaf 1)))))))
>
```

iii

```
#lang racket

(define (PermTreeToPerms T)
  (cases Permtree T
    [root (children) (apply append (map (λ (child) (PermTreeToPerms child)) children))]
    [node (num children) (map (λ (x) (cons num x))
                              (apply append (map (λ (child) (PermTreeToPerms child)) children)))]
    [leaf (num) (list (list num))]))
```

iv

```
(define (PruneSmooth T d)
  (cases Permtree T
    [root (children) (root (map (λ (x) (PruneSmooth x d)) children))]
    [node (num children) (node (getnum T) (map (λ (x) (PruneSmooth x d))
                              (filter (λ (y)
                                        (< (abs (- (getnum y) (getnum T))) d))
                                      children)))]
    [leaf (num) (leaf num)]))
```

v

```
(define (SmoothPerms L d)
  (PermTreeToPerms (PruneSmooth (ListToPermTree L) d)))
```

q.2

```
#lang racket
(require (lib "eopl.ss" "eopl"))

(define-datatype Permtree Permtree?
  [root (children (list-of Permtree?))]
  [node (num integer?) (children (list-of Permtree?))]
  [leaf (num integer?)])

(define (permtree-helper l n)
  (if (null? l)
      (leaf n)
      (node n (map (λ (x) (permtree-helper (remove x l) x)) l))))

(define (ListToPermTree L)
  (root (map (λ (x) (permtree-helper (remove x L) x)) L)))

(define (PermTreeToPerms T)
  (cases Permtree T
    [root (children) (apply append (map (λ (child) (PermTreeToPerms child)) children))]
    [node (num children) (map (λ (x) (cons num x))
                              (apply append (map (λ (child) (PermTreeToPerms child)) children)))]
    [leaf (num) (list (list num))]))

(define (getnum N)
  (cases Permtree N
    [root (children) null]
    [node (num children) num]
    [leaf (num) num]))

(define (PruneSmooth T d)
  (cases Permtree T
    [root (children) (root (map (λ (x) (PruneSmooth x d)) children))]
    [node (num children) (node (getnum T) (map (λ (x) (PruneSmooth x d))
                              (filter (λ (y)
                                        (< (abs (- (getnum y) (getnum T))) d))
                                      children)))]
    [leaf (num) (leaf num)]))
```

```
(define (SmoothPerms L d)
  (PermTreeToPerms (PruneSmooth (ListToPermTree L) d)))
```