テキスト処理 第12回 (2006-07-11)
splitstrレポート説明

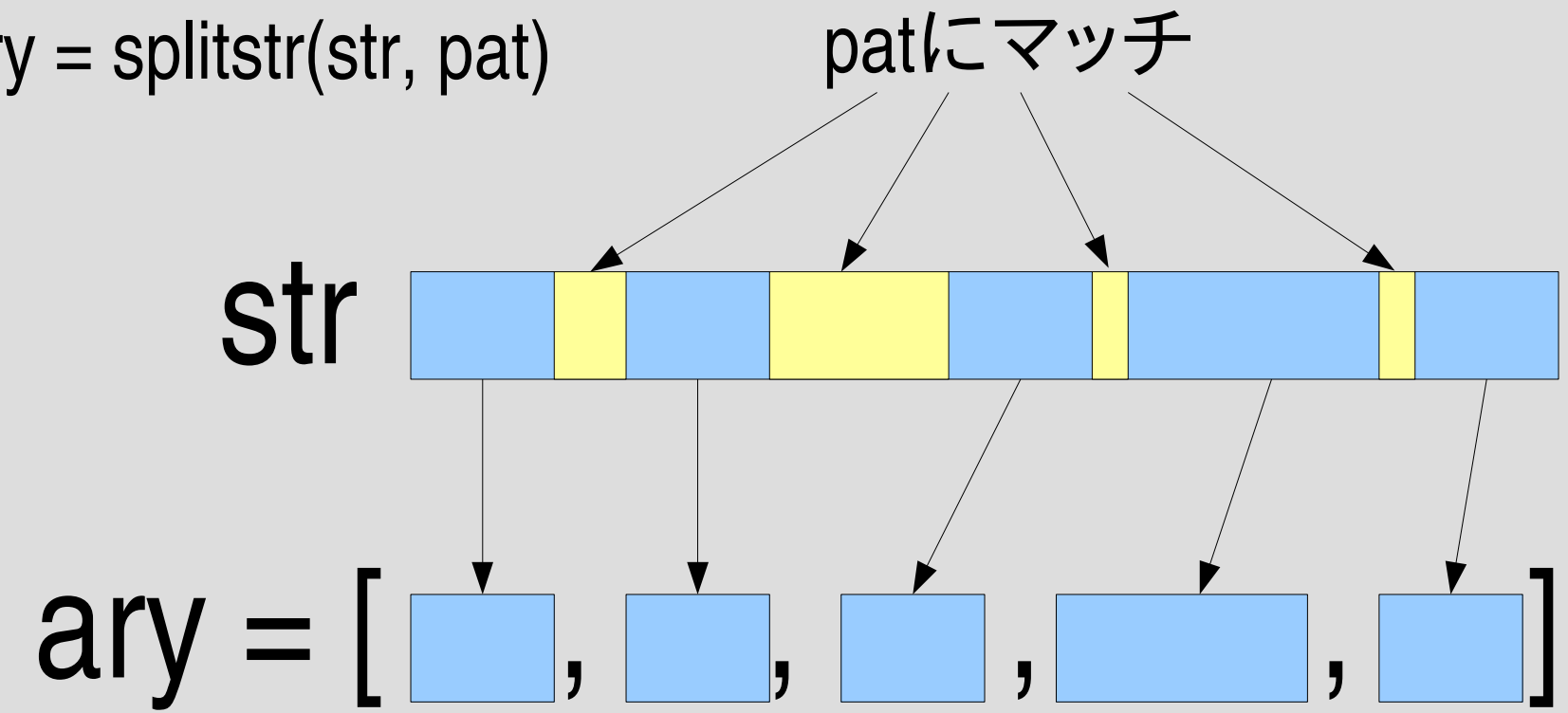
田中哲
産業技術総合研究所
情報技術研究部門
akr@isc.senshu-u.ac.jp
http://staff.aist.go.jp/tanaka-akira/textprocess/

レポート

- splitstr(str, pat) を実装し解説せよ
- splitstr(str, pat) は str を正規表現抽象構文木 pat がマッチする部分で分割し、マッチしない部分を要素とする配列を生成して返す
- 実装したらユニットテストで確認してほしい
- ユニットテストでも不明な挙動があれば String#split で確認する
- ∠切 2006-07-11 16:20
- IT's class
- 拡張子が txt なテキストファイルがよい

レポート

- ary = splitstr(str, pat)



- 注意: マッチの終端の空文字列にマッチした場合、その間の空文字列は配列の要素にならない

レポートのヒント

1. find_match を使う
2. 今回は再帰不要
3. まず空文字列のことを気にせずに書いてみる
4. 先頭と末端の空文字列にはマッチしない

実装例

```
def splitstr(str, pat)
  ary = str.split(//)
  beg = 0
  result = []
  while beg < ary.length
    r = find_match(ary, pat, beg)
    break if !r
    s, e = r
    if beg == s && beg == e
      r = find_match(ary, pat, beg+1)
    end
    break if !r
    s, e = r
    result << ary[beg...s].join
    beg = e
  end
  if beg < ary.length
    result << ary[beg..-1].join
  end
  result
end
```

処理の流れ

```
def splitstr(str, pat)
  初期化
  while 区切りが有る
    区切り以前を結果へ追加
    次の区切りを探す準備
  end
  最後の区切り以降を結果へ追加
  結果を返す
end
```

初期化

```
def splitstr(str, pat)
  ary = str.split(//)
  beg = 0
  result = []
  while beg < ary.length
    r = find_match(ary, pat, beg)
    break if !r
    s, e = r
    if beg == s && beg == e
      r = find_match(ary, pat, beg+1)
    end
    break if !r
    s, e = r
    result << ary[beg...s].join
    beg = e
  end
  if beg < ary.length
    result << ary[beg..-1].join
  end
  result
end
```

区切りが有るあいだ繰り返す

```
def splitstr(str, pat)
  ary = str.split(//)
  beg = 0
  result = []
  while beg < ary.length
    r = find_match(ary, pat, beg)
    break if !r
    s, e = r
    if beg == s && beg == e
      r = find_match(ary, pat, beg+1)
    end
    break if !r
    s, e = r
    result << ary[beg...s].join
    beg = e
  end
  if beg < ary.length
    result << ary[beg..-1].join
  end
  result
end
```

普通の区切りの処理

```
def splitstr(str, pat)
  ary = str.split(//)
  beg = 0
  result = []
  while beg < ary.length
    r = find_match(ary, pat, beg)
    break if !r
    s, e = r
    if beg == s && beg == e
      r = find_match(ary, pat, beg+1)
    end
    break if !r
    s, e = r
    result << ary[beg...s].join
    beg = e
  end
  if beg < ary.length
    result << ary[beg..-1].join
  end
  result
end
```

区切りの右端に幅のない区切りが見つかったときの処理

```
def splitstr(str, pat)
  ary = str.split(//)
  beg = 0
  result = []
  while beg < ary.length
    r = find_match(ary, pat, beg)
    break if !r
    s, e = r
    if beg == s && beg == e
      r = find_match(ary, pat, beg+1)
    end
    break if !r
    s, e = r
    result << ary[beg...s].join
    beg = e
  end
  if beg < ary.length
    result << ary[beg..-1].join
  end
  result
end
```

区切り以前を結果へ追加

```
def splitstr(str, pat)
  ary = str.split(//)
  beg = 0
  result = []
  while beg < ary.length
    r = find_match(ary, pat, beg)
    break if !r
    s, e = r
    if beg == s && beg == e
      r = find_match(ary, pat, beg+1)
    end
    break if !r
    s, e = r
    result << ary[beg...s].join
    beg = e
  end
  if beg < ary.length
    result << ary[beg..-1].join
  end
  result
end
```

次の区切りを探す準備

```
def splitstr(str, pat)
  ary = str.split(//)
  beg = 0
  result = []
  while beg < ary.length
    r = find_match(ary, pat, beg)
    break if !r
    s, e = r
    if beg == s && beg == e
      r = find_match(ary, pat, beg+1)
    end
    break if !r
    s, e = r
    result << ary[beg...s].join
    beg = e
  end
  if beg < ary.length
    result << ary[beg..-1].join
  end
  result
end
```

最後の区切り以降を結果へ追加

```
def splitstr(str, pat)
  ary = str.split(//)
  beg = 0
  result = []
  while beg < ary.length
    r = find_match(ary, pat, beg)
    break if !r
    s, e = r
    if beg == s && beg == e
      r = find_match(ary, pat, beg+1)
    end
    break if !r
    s, e = r
    result << ary[beg...s].join
    beg = e
  end
  if beg < ary.length
    result << ary[beg..-1].join
  end
  result
end
```

結果を返す

```
def splitstr(str, pat)
  ary = str.split(//)
  beg = 0
  result = []
  while beg < ary.length
    r = find_match(ary, pat, beg)
    break if !r
    s, e = r
    if beg == s && beg == e
      r = find_match(ary, pat, beg+1)
    end
    break if !r
    s, e = r
    result << ary[beg...s].join
    beg = e
  end
  if beg < ary.length
    result << ary[beg..-1].join
  end
  result
end
```

中身について

- 幅のないケースの扱いは面倒臭い

ざっと眺めた結果

- できている (ユニットテストが通っている) 人もいなくはない
- 空文字列にマッチしない場合までであればかなりできている
- 空文字列にマッチする場合は難しすぎたらしい

教訓

- 空文字列にマッチするパターーンは問題を引き起こす
  - :rep で無限再帰
  - gsub で無限ループ
  - split でいろいろ
- 開発者の対策: 常に空文字列を考える (境界条件重要)
- ユーザ側の対策: 空文字列にマッチするパターーンは必要でない限り書かない