A linear time algorithm for ranking scores with ties

Kjell Post
Irsta Fotostudio
kjell@irstafoto.se

Abstract

In competitions, it is useful to present a list of winning entries and their scores, but in case of several entries with the same score we must present not only their final place, but also if their place was tied. This memo describes an $O(n)$ algorithm to rank the entries so that each entry has a place and a tie marker, e.g., "2nd place" or "Shared 1st place".

1 The ranking algorithm

Consider a competition with a set of scores S, assumed sorted:

$$S = \{0, 0, 10, 75, 76, 76, 76, 77, 77, 77, 78, 79, 80, 80\}$$

When the scores are presented, we would like to display them as follows:

Shared	13th place:	0
Shared	13th place:	0
	12th place:	10
	11th place:	75
Shared	8th place:	76
Shared	8th place:	76
Shared	8th place:	76
Shared	5th place:	77
Shared	5th place:	77
Shared	5th place:	77
	4th place:	78
	3th place:	79
Shared	1st place:	80
Shared	1st place:	80

We assume S can be viewed both as a set and as a list $S[0] \dots S[N-1]$ where N is the number of scores. The following algoritm annotates each score with attributes *place* (denoting the score's final place) and *tied* (a boolean, denoting if the place is tied).

```
procedure rank(S):
    N ← length(S)
    if N = 0 return
    p ← 1
    topScore ← ∞
    for k ∈ {N − 1, N − 2, ... 1, 0} // Visit each score in ascending order
        if S[k] < topScore          // A new distinct score ⇒ a new place
            p ← N − k
            topScore ← S[k]
        S[k].place ← p
        S[k].tied ← false
        if k < N − 1 and S[k + 1].place = p // If previous score is the same, we have a tie
            S[k].tied ← true
            S[k + 1].tied ← true
```

2 Python implementation

```
#!/usr/bin/env python

photos = [
     { 'score':  0 },
     { 'score':  0 },
     { 'score': 75 },
     { 'score': 76 },
     { 'score': 76 },
     { 'score': 76 },
     { 'score': 77 },
     { 'score': 77 },
     { 'score': 77 },
     { 'score': 78 },
     { 'score': 79 },
     { 'score': 80 },
     { 'score': 80 },
     { 'score': 10 },
]

# sort photos by score
photos.sort(lambda x,y : cmp(x['score'], y['score']))

def rank(s):
    N = len(s)
    if N == 0:
        return
    p = 1
    topScore = 999999
    # visit s[N-1], s[N-2], ... s[1], s[0]
    for k in range(N-1, -1, -1):
        if s[k]['score'] < topScore:
            p = N - k
            topScore = s[k]['score']
        s[k]['place'] = p
        s[k]['tied'] = False
        if k < N-1 and s[k+1]['place'] == p:
            s[k]['tied'] = True
            s[k+1]['tied'] = True

rank(photos);
for p in photos:
    if p['tied']:
        print 'Shared %4s place: %3s' % (p['place'], p['score'])
    else:
        print '       %4s place: %3s' % (p['place'], p['score'])
```