
Subject: Re: Fast way to find the same value in an array
Posted by [greg michael](#) on Sun, 09 Jul 2006 21:56:28 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi Boto,

Try this. On my (very ordinary) machine, it takes about 10 secs to search 1e5, several minutes for 1e6. Didn't try 1e7, but I'd guess not more than a few hours...

```
function MakeGalaxies,n
p=replicate({x:0.,y:0.,z:0.},n)
seed=0
p.x=randomu(seed,n)*1000.
p.y=randomu(seed,n)*1000.
p.z=randomu(seed,n)*1000.
return,p
end
```

```
function distance,p1,p2
return,sqrt((p2.x-p1.x)^2+(p2.y-p1.y)^2+(p2.z-p1.z)^2)
end
```

```
pro search,p,dist
;recursively splits the search volume into n_split^3 subvolumes. When
there are fewer than 'threshold' points
;in a subvolume, checks for matches the brute force way - every point
against every other.
```

```
n_split=3
threshold=75
```

```
s=(size(p))[1]
```

```
if s gt threshold then begin
xmn=min(p.x,max=xmx)
ymn=min(p.y,max=ymx)
zmn=min(p.z,max=zmx)
```

```
xg=xmn+findgen(n_split+1)*(xmx-xmn)/n_split ;grid boundaries
yg=ymn+findgen(n_split+1)*(ymx-ymn)/n_split
zg=zmn+findgen(n_split+1)*(zmx-zmn)/n_split
```

```
for j=0,n_split-1 do begin
for k=0,n_split-1 do begin
for l=0,n_split-1 do begin
w= where( (p.x ge xg[j]-dist) and (p.x lt xg[j+1]+dist) and $
(p.y ge yg[k]-dist) and (p.y lt yg[k+1]+dist) and $
```

```

        (p.z ge zg[l]-dist) and (p.z lt zg[l+1]+dist) )
    if n_elements(w) ge 2 then begin
        res=search(p[w],dist)
    endif
endfor
endfor
endfor

endif else begin
for i=0,s-2 do begin
for j=i+1,s-1 do begin
if distance(p[i],p[j]) le dist then begin
print,p[i],p[j]
endif
endif
endif
endif

endelse

end

```

IDL> p=makegalaxies(1e5)

IDL> search,p,.5

```
{ 454.665 175.794 87.0097}{ 454.816 175.504
87.3471}
```

```
{ 556.761 981.076 933.809}{ 556.654 980.922
934.065}
```

```
{ 981.340 196.286 105.551}{ 981.387 196.102
105.703}
```

IDL> p=makegalaxies(1e6)

IDL> search,p,.2

```
{ 330.280 196.344 99.2760}{ 330.297 196.277
99.4484}
```

```
{ 249.980 173.201 640.821}{ 250.142 173.111
640.774}
```

```
{ 62.5799 42.0245 779.031}{ 62.4897 41.8944
779.097}
```

```
{ 297.023 425.521 640.250}{ 296.952 425.609
640.149}
```

```
{ 98.4379 461.400 673.197}{ 98.4014 461.355
673.278}
```

```
{ 189.966 444.613 924.605}{ 190.002 444.629
924.567}
```

```
{ 161.315 726.953 96.7764}{ 161.361 726.784
96.7790}
```

```
{ 254.492 823.492 494.062}{ 254.499 823.541
494.128}
```

{ 143.837 892.034}	736.945	891.959}{	143.846	737.073
{ 548.274 158.257}	302.271	158.284}{	548.251	302.408
{ 638.083 240.530}	100.108	240.606}{	638.107	100.159
{ 628.376 191.481}	117.708	191.394}{	628.338	117.743
{ 569.059 375.896}	497.433	375.822}{	569.195	497.334
{ 519.456 547.015}	995.505	547.080}{	519.352	995.619
{ 426.479 849.277}	719.818	849.407}{	426.415	719.915
{ 824.665 151.754}	139.718	151.656}{	824.762	139.576
{ 873.701 222.147}	398.928	222.038}{	873.774	398.952
{ 924.378 62.9983}	468.693	62.9666}{	924.546	468.719
{ 858.907 717.239}	486.427	717.181}{	858.761	486.428
{ 720.754 405.215}	820.070	405.199}{	720.787	820.032
{ 781.977 563.536}	897.380	563.545}{	782.112	897.508

Seems that splitting into 3x3 is better than 10x10. I suspect this is something to do with how the 'where' expression works. I don't know much about IDL timing.

Meanwhile, France won...
regards,
Greg
