Subject: Modifying an array while conserving memory
Posted by Randall Skelton on Fri, 24 May 2002 00:12:55 GMT
View Forum Message <> Reply to Message

Hi all,

I have a large array and I would like to 'insert' some data into the
middle of it. Imagine an array of 1000 points and having 100 points to
insert beginning at index 500 (the resulting array will have 1100 points).
Typically, I do not know the length of data I wish to insert until after
'a' is defined.

```
a = findgen(1000)
b = randomu(seed,100)
c = fltarr(1100)        ; seems wasteful to use more memory
c[0:499] = a[0:499]
c[500:599] = b
c[600:1099] = a[500:999]
```

In reality, 'a' is of order 2e7 so I would like to avoid making
multiple copies of it. Does anyone have any suggestions regarding the
most memory efficient way of doing this?

Many thanks,
Randall

Subject: Re: Modifying an array while conserving memory
Posted by Randall Skelton on Fri, 24 May 2002 13:20:10 GMT
View Forum Message <> Reply to Message

In for an inch, in for a mile... After reading the EDG it is now glaringly
obvious why my naive 'realloc' attempt was doomed. It seems IDL provides
access to the dymanic memory allocation routines as IDL_MemAlloc() and
IDL_MemFree(). Of course, the one they forgot to write about is the
function IDL_MemRealloc-- the symbols for which can be found in libidl.so.

Cheers,
Randall

On Fri, 24 May 2002, Randall Skelton wrote:

> It seems this will be my 3rd feature request of the week as my long-shot
> attempts at using realloc on an IDL passed array have failed miserably :(

Subject: Re: Modifying an array while conserving memory

Posted by Xiangyun Qiu on Fri, 24 May 2002 15:38:46 GMT
View Forum Message <> Reply to Message

Randall Skelton wrote:
Hi there,

Provided TEMPORARY() function works as we expect, I think the following steps can
use less memory:

```
a = findgen(1000)
b = randomu(seed,100)
a = shift(TEMPORARY(a), 500)
c = [TEMPORARY(b),TEMPORARY(a)]
c = Shift(TEMPORARY(c), 500)    ; here happens to be 500 too, because
500 = 1000 - 500
```

This solely rely on TEMPORARY function, live or die with it.

Xiangyun

```
>
>  Hi all,
>
>  I have a large array and I would like to 'insert' some data into the
>  middle of it.  Imagine an array of 1000 points and having 100 points to
>  insert beginning at index 500 (the resulting array will have 1100 points).
>  Typically, I do not know the length of data I wish to insert until after
>  'a' is defined.
>
>  a = findgen(1000)
>  b = randomu(seed,100)
>  c = fltarr(1100)        ; seems wasteful to use more memory
>  c[0:499] = a[0:499]
>  c[500:599] = b
>  c[600:1099] = a[500:999]
>
>  In reality, 'a' is of order 2e7 so I would like to avoid making
>  multiple copies of it.  Does anyone have any suggestions regarding the
>  most memory efficient way of doing this?
>
>  Many thanks,
>  Randall
```

--
Xiangyun

Subject: Re: Modifying an array while conserving memory
Posted by Randall Skelton on Fri, 24 May 2002 15:43:27 GMT
View Forum Message <> Reply to Message

Well... after a few insightful words from a shy fellow at RSI, it seems
what I have been trying to do is absolutely impossible :(

Thanks to all who offered comments.

Randall

On Fri, 24 May 2002, Randall Skelton wrote:

> In for an inch, in for a mile... After reading the EDG it is now glaringly
> obvious why my naive 'realloc' attempt was doomed. It seems IDL provides
> access to the dymanic memory allocation routines as IDL_MemAlloc() and
> IDL_MemFree(). Of course, the one they forgot to write about is the
> function IDL_MemRealloc-- the symbols for which can be found in libidl.so.
>
> Cheers,
> Randall

Subject: Re: Modifying an array while conserving memory
Posted by Pavel A. Romashkin on Fri, 24 May 2002 16:11:43 GMT
View Forum Message <> Reply to Message

I don't think this takes care of it. Temporary does not work like this.
It is useful for modifying memory contents in place, but not for
concatinating memory allocations together.
Pavel

Xiangyun Qiu wrote:
>
> Randall Skelton wrote:
> Hi there,
>
> Provided TEMPORARY() function works as we expect, I think the following
> steps can
> use less memory:
>
> a = findgen(1000)
> b = randomu(seed,100)
> a = shift(TEMPORARY(a), 500)
> c = [TEMPORARY(b),TEMPORARY(a)]
> c = Shift(TEMPORARY(c), 500) ; here happens to be 500 too, because
> 500 = 1000 - 500
>

> This solely rely on TEMPORARY function, live or die with it.

Subject: Re: Modifying an array while conserving memory
Posted by Xiangyun Qiu on Fri, 24 May 2002 17:06:46 GMT
View Forum Message <> Reply to Message

I was hoping this would only use memory size of about size(a) +
2*size(b) during
operation; when it's done, only size(a) + size(b). If the conccatination
causes the
problem, I currently have no solution to this.
Thanks,

Xiangyun

"Pavel A. Romashkin" wrote:
>
> I don't think this takes care of it. Temporary does not work like this.
> It is useful for modifying memory contents in place, but not for
> concatinating memory allocations together.
> Pavel
>
> Xiangyun Qiu wrote:
>>
>> Randall Skelton wrote:
>> Hi there,
>>
>> Provided TEMPORARY() function works as we expect, I think the following
>> steps can
>> use less memory:
>>
>> a = findgen(1000)
>> b = randomu(seed,100)
>> a = shift(TEMPORARY(a), 500)
>> c = [TEMPORARY(b),TEMPORARY(a)]
>> c = Shift(TEMPORARY(c), 500) ; here happens to be 500 too, because
>> 500 = 1000 - 500
>>
>> This solely rely on TEMPORARY function, live or die with it.

--
Xiangyun
