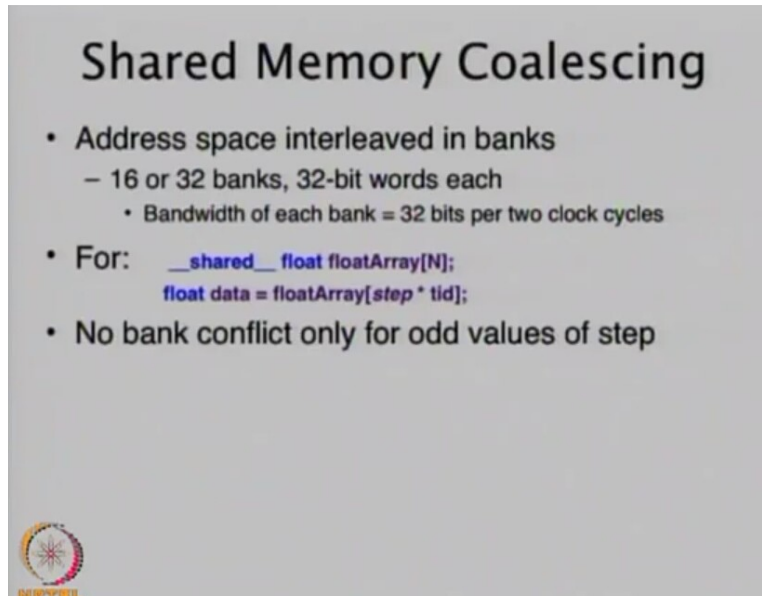


Parallel Computing
Prof. Subodh Kumar
Department of Computer Science & Engineering
Indian Institute of Technology – Delhi


Module No # 06
Lecture No # 27
CUDA (Contd.)

(Refer Slide Time: 00:25)



Shared Memory Coalescing

- Address space interleaved in banks
 - 16 or 32 banks, 32-bit words each
 - Bandwidth of each bank = 32 bits per two clock cycles
- For: `__shared__ float floatArray[N];`
`float data = floatArray[step * tid];`
- No bank conflict only for odd values of step



So, give me an example I will repeat it when I ask about the specific thing again so but it is not efficient getting the intake was plot stored an array of size of 32 + think of a block because block as a same structure you know what an array of size 32 and I read that array and it results in the Closets memory axis. For example, global memory but on the shade memory side its is inefficient because it is blocked there is a bank conflict.


So in general when you declare an array of certain size .And you say I am going to access using some multiple every in its based on TID every third it is not always you access things in one particular it is not always the first third axis is the 0 and so on for example XYZ and I want to read X so X is in every third so I will say I have got an array of floats and I am reading 3 times my tid as long as this time TID is an odd number will be no conflicts and go convince yourself with that.

If 3 is 2 3 is not 2. if that multiplier is 2 what will happen it goes 2, 4, 8 after 14 the address will come back to 0 16 is in 0 track for 1X and for 2 that again is 32 banks again it is 32 times something so as long is that is odd there is no conflict.

(Refer Slide Time: 03:09)

Shared Memory Coalescing

- Address space interleaved in banks
 - 16 or 32 banks, 32-bit words each
 - Bandwidth of each bank = 32 bits per two clock cycles
- For: `__shared__ float floatArray[N];`
`float data = floatArray[step * tid];`
- No bank conflict only for odd values of step
- Conflict for: `__shared__ char charArray[N];`
`char data = charArray[i * tid];` (Not for 2.x)



If your reading a character array then what is going to happen first says is I am going to read something from zeroth location of the byte array the first one says begin from the first location of the first array second says second side of the second array and the first four are reading from the same bank 4 bytes are in the same its 32bit cycle.

So in 1 dot X always in conflict because in basically you reading from same bank that says your conflicting. in 2 dot X should it be conflicting in 2 dot it does not but why we always say a word transaction 32 width so that is not the reading for different things from that bank one thing from that bank all for take down section their byte from it ok so it would not conflict. What about this?

(Refer Slide Time: 04:36)

Shared Memory Coalescing

- Address space interleaved in banks
 - 16 or 32 banks, 32-bit words each
 - Bandwidth of each bank = 32 bits per two clock cycles
- For: `__shared__ float floatArray[N];`
`float data = floatArray[step * tid];`
- No bank conflict only for odd values of step
- Conflict for: `__shared__ char charArray[N];`
`char data = charArray[i * tid];` (Not for 2.x)
- Resolve with array of struct (of four chars), or:
`char data = charArray [4 * tid];`



Right that is not my question so in 1 dot X you can make bigger array byte and resolve this conflict and it is not too relevant because byte arrays are not conflict anymore but what about double arrays if I am reading spit double array is an 8 byte. So I am reading 8 bits at a time. what is going to happen?


Your reading from 2 banks 16 people are reading so first two and second two next two that is all fine but what happens to the next the second half or second quarter or goes back to the first only half of them can be provided the data at one time. What would this access do for global memory not a problem its it doubles 8 times 16 rite 120 bytes one memory transaction will take place. there is no notion of conflict.in 2 dot X it also there is no conflict.

(Refer Slide Time: 06:23)

Shared Memory Coalescing

- Address space interleaved in banks
 - 16 or 32 banks, 32-bit words each
 - Bandwidth of each bank = 32 bits per two clock cycles
- For: `__shared__ float floatArray[N];`
`float data = floatArray[step * tid];`
- No bank conflict only for odd values of step
- Conflict for: `__shared__ char charArray[N];`
`char data = charArray[i * tid];` (Not for 2.x)
- Resolve with array of struct (of four chars), or:
`char data = charArray [4 * tid];`

How about `doubleArray`? Conflict only for 1.x




In 2 dot X conflict is only for 1 dot X with in half work a 32 banks now so what will happen the first ill read the first pair the second will read the second pair and 16 will read from 32 so there is no rap around that is necessary okay. So now let us review something we have already talked about.

(Refer Slide Time: 06:53)

Block Sizing

- Have more blocks in the grid than #SMs
- Have enough threads in a block to fill one SM
 - Multiple blocks in an SM is another possibility
 - Use >1 to fill cycles during sync of one block
 - \Rightarrow `grid-size $\geq 2 \times$ #SMs` (Example: 96 block grids)
 - Resource per block must be small enough to fit SM
- Block size should be multiple of warp size
 - Rule of the thumb: 192 or 256 thread/block
 - Never fewer than 64



We talk about the factor that we need block as bigger as possible so that there are enough warps at the same time you cannot make blocks bigger bigger and bigger there is a limit resource. For example, you shared memory in every thread of the block probably needs the share of the shared memory block becomes bigger it probably need more warps and similarly if there are more

thread you need more registers and so on. We talked about the fact that you can make each thread do more which is equivalent in some ways to saying that block is small effectively you're making a block big it is said to be small but this block is doing more indiscriminate the block is big.

Can you do that all the way let the block be 32 sized and everything is getting done within the thread no right why? again its memory is a some of an tragical issue because all those all that work had to be done if it is in the global memory then all these threads this thread is reading 100 items from the global memory and is the same thing 100 different threads reading one item in global memory.

But we need many wars to be active right why do we need many wars to be active not just memory relevancy but also relative relevancy and that in not that not their just go that may come later. Instruction latency is order of 20 to 24 clock cycle for more instructions so why is that happening if the next instruction needs the result of the instruction. Then we cannot schedule it some body has to of course memory has the same thing if you need data from the memory immediately and you have to wait until it comes.

While you work you have to the general rule you want the block size to be 32 x something it is generally a good idea to start something like 256 hertz in a block if you make it too small. Why not to make a block small I have lots of blocks but each block is small same question why not to make 32 x because then inter block synchronization is not possible you want to make blocks as big as possible so that they can communicate each other.

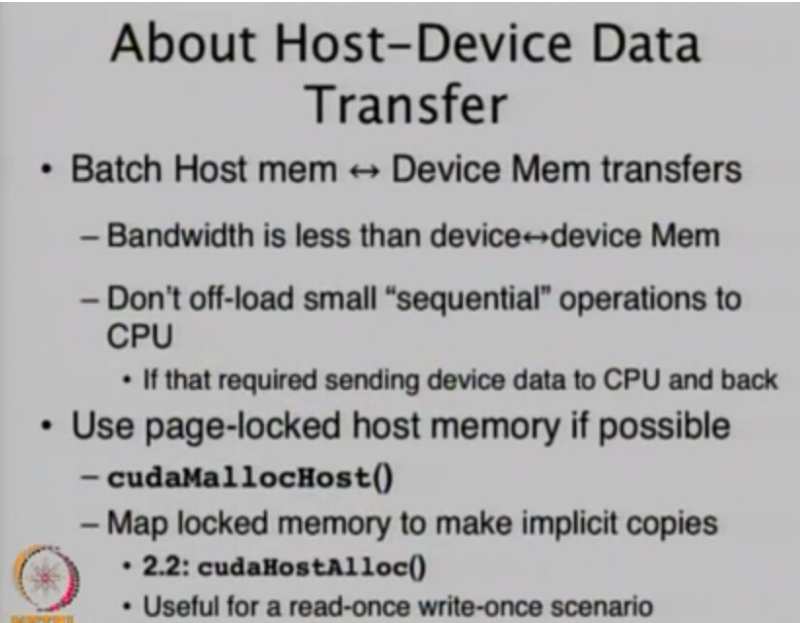
Make it as small as possible so that it does not depend on each other probably a sweet part somewhere which is greater than 32 and less than 1024? No if you can lessen the dependency between blocks its good but you would not be able to do that we talked about in fact that reviewed will also come we talked about the issue of parallelism. How do you make thing parallel sometimes make things more parallel by repeating the computation that somebody has done.

Instead of taking that guy has done the result you just do your own and repeat the computation when you do that when your not depending on some body else get rid of synchronization they need to be in same block you just compute your own other wise you have to sync the threads at that point so that you know that guy result is not ready in that grid whatever that fellow is produced so you should you can get rid of synchronization but just repeating things or re organizing your algorithm.

If you can good if you cannot then you have to make a block bigger. maximum sized as blocks get bigger that take more resources. more over what is going to happen is that a eventually blocks will innovatively needs some sync that is why that block their sharing something dependently that guy to produce some thing to use that later on whenever you have such synchronization in the block basically slowly the blocks start reaching that point and start becoming inactive.

I have reach that point I have to wait for others to reach that point and I cannot do anything until everybody has so that block works eventually get out of commission for a little while and if you have nothing else to run then you are going to go ideal and that is why you do not want blocks to be too big and then are also issue resources. If you make it too big you may not make it too fit all the memory that needs.

(Refer Slide Time: 13:32)



About Host-Device Data Transfer

- Batch Host mem ↔ Device Mem transfers
 - Bandwidth is less than device↔device Mem
 - Don't off-load small "sequential" operations to CPU
 - If that required sending device data to CPU and back
- Use page-locked host memory if possible
 - `cudaMallocHost()`
 - Map locked memory to make implicit copies
 - 2.2: `cudaHostAlloc()`
 - Useful for a read-once write-once scenario

Ok in terms of memory transfers your take copies typically you would see this a copy stuff into the CUDA device you do the computation start out but you want to do as copy all the stuff in one call so you batch memory transfers as much as possible and more than that if you can keep the memory in the GPU device want to keep it there not to bring it back even if you need sometimes you need synchronization across blocks and you have to start new content but that doesn't mean you need to bring the memory back.

And if you can organize your computation in a way if you compute something end the kernel whatever you compute still in the global memory you start the new kernel and just start from the same place from where ever you have computed start reading that partial results from previous and kernel start computing start working on next content moreover.

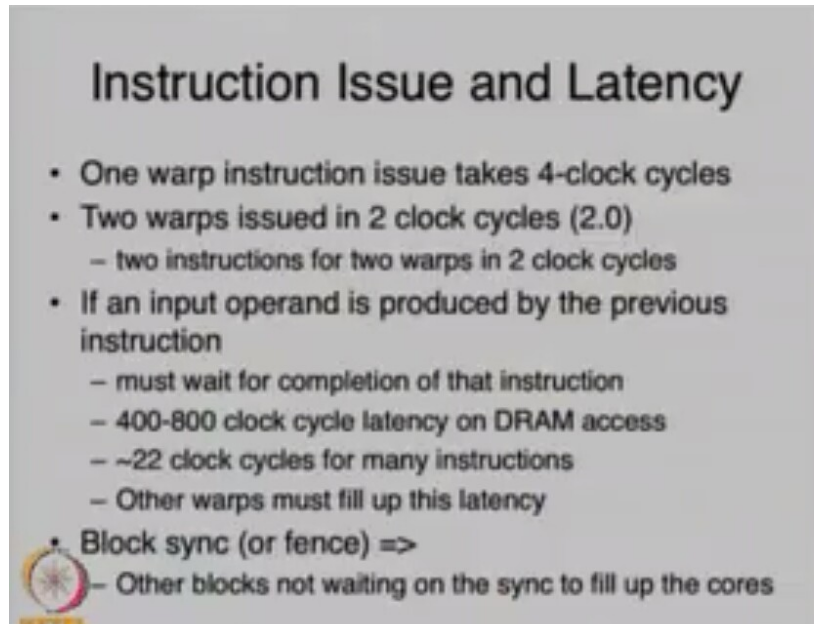
Sometimes you would say sequential part of the code I am going to do this on the GPU use on the same if my thread is 0 to this otherwise wait and waste lot of GPU resources the alternative is you stop that kernel come back on a CPU thread you do the sequential part and start the kernel again this only helps if the sequential part is really really big that sequential part was relatively small, small loop or 5, 6, 10 lines of code you just repeat it on every thread if you have to make out the thread ideal into one of the thread to the sequential part .

So that you do not want to start kernel and that copy memory that back and fourth and also in general if you can map we talk about the map that rest on CPU and GPU so that you write on a CPU and read it on the GPU if you can do that you can choose that special if you really have this past meaning that you want to do needs all the rights at the beginning on the host do the kernel and reads on the host.

You are not interspersing needs and rights then you want to do it in the mapping. There is a system thread fact so GPU you have to be careful about that and you have to use it. GPU to CPU is lower something wrong with that result. The reason is more historic typically graphic pipeline works in the forward direction and it is always optimized in that way and so in order to bring something back you have to clear some stages of pipeline and stall now you get that data

back in many stages of architecture that part has made you have to flush stages of pipe line to bring the data back.

(Refer Slide Time: 17:29)



Instruction Issue and Latency

- One warp instruction issue takes 4-clock cycles
- Two warps issued in 2 clock cycles (2.0)
 - two instructions for two warps in 2 clock cycles
- If an input operand is produced by the previous instruction
 - must wait for completion of that instruction
 - 400-800 clock cycle latency on DRAM access
 - ~22 clock cycles for many instructions
 - Other warps must fill up this latency
- Block sync (or fence) =>
 - Other blocks not waiting on the sync to fill up the cores

We talked about the cash line size being to 56 bytes which is actually 120 bytes right that is how much maximum you can read in one short it also means I said that data reads can be 32, 64, 28 bytes from the memory it is always 120 bytes if it is cached which means if you need a small section of it you are doing false sharing or you may be doing false sharing.

Ok back to this the numbers were talking about you take 4 clock cycles to launch all the threads of a given walk on 1 dot X it processes. There are only 8 processors and you have too 32 to run so do it over for a 4 clock cycles okay. It is now with 2 dot X two half you can launch it every clock cycle there is two halves on a launch and each has 16 processors effectively 32 processors to run and so 32 threads can belong in parallel.

The truth is in half fashion at different half work launched on a second side. On a second side group of 16 processors well as your half are longs on the first launch on first and your second half and so on. I actually do not know the answer to that it might have been a more smoother transition from the architecture point of view from the old architectural point of view to the next but I really do not know.

Made it faster instead of four clock cycles do it over two clock cycles but I do not know what logic might have been but it can be seeking of many reasons. One of the weakness of CUDA is if else right when ever branching happens there is prize to pay so then we have figured that branch all the bench mark have that branches tend to be slow.

And if you make all the 32 clocks run at a same time you are more likely to be wasting a lance because its not always in fact may be in those bench marks is rare that all 32 at a same time. So then it is a lead so that you can break it into two that may be one breezing but I have truly do not know that a pure explanation.

To get data from the ram it takes from 400-800 clock cycles and it depends on the rate coming from the actual memory for most instructions as I said 20 to 24 clocks meaning that. Now let us talk from 2.0 point of view every clock you can launch one instructions and the your instruction you launched is 22 clock cycles forget about read for the time whatever instruction you have general $A + B$ register 1 + register 2 which means for the next 22 clock cycles you need result surface instruction.

You are out of commission every work without a commission for 22X is an instruction count and for 100 instructions it runs 100 clock but 2100 clocks its actually going to be visiting actually doing nothing see need that many extra work just to fill will even if your not doing with it axes to the minimum number of clocks you get in that order 22 probably 32 is a better number.


And every time if it goes to read now its going to wait up to 800 clocks and you need 800 blocks that is the main ok but that is the level of parallelism your going to meet because you do not need 800 clocks because nobody said you need to run in you know every walk can be run every 22 clock cycles and if you have to wait for 800 clocks then other works can be run 22 clock cycles.

They do not have read write instructions so there is a notion of your compute density is ratio of how many instructions you have of 22 cycles variety you have 800 cycle variety the needs vs computes and you have to have at least that many walks and probably many times for because many of then will also have down types they cannot run their depending on something else.

(Refer Slide Time: 24:37)

Memory Optimization

- Minimize data transfer
- Keep data and their computation on the device
- Batch transfers
- Using page-locked host memory (if possible)
- Use coalesce-able memory access
- Use mapped memory
 - if device reads and writes only once




Ok we talked about memory so some the other things you probably need to be careful about a one very often you will realize that you have local variables automatic variables auto variables in different local method they taking 800 clock cycles what would have been talking around one clock cycle and you probably want to especially careful about and you want to check that you are not going into thread local memory.

(Refer Slide Time: 25:13)

Thread-Local Memory

- Auto variables placed in local memory are:
 - Arrays not const indexed
 - at least not determined at compile time
 - Large structures or arrays
 - Too many local live variables (register spilling)
- nvcc reports total local memory usage
 - use option `--ptxas-options=-v` option
- Inspect `cubin` object file using `cuobjdump`



And there are ways to check the PTX compiler and the options tell you as it turns out PTX compiler is not end all at the end PTX compiler code which is architecture independent is further compiled and if your running on older app architecture then PTX is guesses but how many registers you may have in this case actually you are double sure you inspect the binary object to see whether local memory is used.

It is possible right to see you done would has taken let me tell you where ever it will give you the assembly language dumbbed and you can do a grid see something dot local and you know that things are going in to local memory. Will never go in local memory its not always possible if you have so many live registers in your code. Yes, in fact I think that should be there by default giving that warning and for some reason that is not their but you can get this information out okay. So I am going to stop here this is basically that we are going to discuss in CUDA class.

On the source file That is about you should report in fact that should have not never heard not that I have done myself but reproduceable every line that definitely did you try to do it on different machineries if possible something wrong in that and that is a problem you should report. Let us get back to the algorithm about that I will show you what that algorithm switching back to continuation of algorithm section that originally starting with this is the question from today's quiz.


(Refer Slide Time: 28:45)



(Refer Slide Time: 28:46)

First 1 problem

- Input: n-bit vector
- Output: minimum index of a 1-bit
- Algorithm:
 - Divide into \sqrt{n} blocks of \sqrt{n} bits each
 - For block i , $B[i] = 1$ if there is a 1 in the block
 - Arbitrary CRCW, $O(n)$ work, $O(1)$ time
 - Find the first j such that $B[j] = 1$
 - $O(n)$ work, $O(1)$ time
 - Find first 1 in the j^{th} block
 - $O(n)$ work, $O(1)$ time



3

We want to find in a vector of 10 one big vector one big element in this vector which is the first this is not hard too if I told you you can have n squared you can have n squared work one that every 1 or 0 at every vector position and I have a want to know which is a first one? 1 width with sequence example for every element I want to know the first one. Know if am the first one I am going to check any one on my left

End processes assigned to me then I can figure out a if any one on my left side how I can figure out on my left so I only processes I is only mine its other minus I is extra I processes one allocate to each position want to know you are the one you are a one and one of you is you are a one that and all the I other processors allocated to me can tell me whether I am the first one or not.

If I am the first one then I am how many people can raise their hands there is only first one so what are N and what are one-time n squared work is safe out. How do you do this if you want to do this in N time without an only one time still widen to two 10 blocks size each and each of these blocks now you can do the imparallel right. I can assign. we may not need that right if we can do this in order work one time then almost like one level of casketing.

But in this case, you do not need full casketing if it is one in all the stage processors could not be one. You have root n processors for root n blocks. Index of first one there are root n processors

counting for root n blocks right one of them is going to succeed so you have to write the larger value.

In order N times constant amount of work, we can say which is the first one how does it help us with finding the prefix minimum. How does this help us finding the prefix out of the if where to try think back and what do you need and did I say what are targeted a so a target is time although we can get a better to complicated long long again time with a work.


Here it was a matter of finding only one right root n long and root N and you can figure out odd off weather are in but you do not get facility over their what you have figured is prefix of root N . Dividing root like we are here and how do you get prefix minimum from root n . there if you do root n squared then it is N root N . we need prefix minimum for every element.

We are going in right direction but it would not lead to same efficiency it will bring it down. So let us enough discussion and enough ideas we have and will see some of them used here also. Let us go through I have mentioned this secondary problem also and is going to used a kind of sub routine here but it is also something that will be and for many different algorithms.

(Refer Slide Time: 39:11)

All Nearest Smaller Value

- For each element $A[i]$ in a vector
 - find the largest j , such that $A[j] < A[i]$
- Algorithm:
 - Find left match:
 - For each i
 - For each $j < i$, $B[i,j] = (A[j] < A[i])$
 - $O(n^2)$ work
 - For each $B[i]$ find largest $j < i$ with $B[i,j] = 1$



4

And this problem is called all nearest smaller value meaning that for every element I got an array of and for every element have to find the nearest to my left or right does not matter nearest to my left which is less than or greater than so you can formulate one for different ways so let's for the concrete I have got an array and I want to know for every position I want to know the nearest element that is less than element to my left .

In fact that is what is shown over in the slide basically say again for one element you have signed a processor I processor for I the element I want to know from the I processor will tell which one of them is less right I want to find the nearest one so one that is less and remaining people write a 0 and I am looking for the first one on my left that given I processors I can do it in auto driver at one time for everybody doing it in parallel not much across the elements.

So every element has n processors or I processors risk order I works one time it is going to determine a its nearest left ok and everybody does that and n times this I and $I_1 + I_2 + I_3$ and so on to the N square work at one time you can find your nearest left ok how does it help us to this by the way is some computative to the algorithm is just described n square time you can find the prefix minimum.

This only finding what prefix again for every element I want to know the smallest of people on my left how is this related to the previous thing problem. All nearest value even more direct connection not minimum to it is the value less than me the closest some of the value not all of the thing there are some of them more than me and some of them are less than me among the one that are less than me and the first one. It is sorted in decreasing order and the next element will be your all nearest very body next element will be the nearest neighbor.

There is more direct connection actually on the slide actually not going to jump but if you can it can be done in parallel. there is no go on business at all. Definition is I have got something on the left I am going to find every bodies nearest ever all nearest smaller value one of them is not going to have anybody on the left that's the minimum. Every bodies are not regression if you think of a sequential process then but in parallel I can throw a bunch of processors there is a

dependence between that guy and this guy who ever say there is nobody on my left widths limitation do not even wait for other people answer.

So there is no regression cause that lead to one when your doing a sequential computation but in parallel it is not necessary the dependence is not their ok just.

(Refer Slide Time: 45:34)

Prefix Minima

- For each element $A[i]$ in a vector
 - find the minimum of all $A[j]$, $j < i$
- Algorithm:
 - Find ANSV
 - For each $A[i]$
 - For each $A[j]$, $j < i$
 - $\text{PrefixMin}[i] = A[j]$ where j is the largest index such that $A[j]$ has no smaller value on its left

•• (m+) (m+) (m+) (m+) (m) (b) (a)

5

For instant a is A element interested in and M is the element that has no body on its left which is smaller than it when it is computed all near smaller values it got itself null depends on actually how you implement but it knows either because it found its own value what if there is a element B in the middle which is smaller than m could it be.

I am asking is that element a possible where B is the minimum but do we know B has and ANSB has a neighbor. any element right of the element found not be the prefix minimum for me. winning smallest element on my left it has a smaller element if B were the smaller element then I could have not so anybody to the right of M has a smaller element so it is not possible one of them is a biggest element.

Can anybody to the left of m be my prefix minimum if anything to the left is smaller than M would have found it when your doing nobody is smaller to my left. So the left most element which do not has ANSV is your prefix minimum for anything to the right of B in the side of

element in the side of B a could not possible to the prefix minimum B would have been a prefix minimum because B have a smaller value to the left which means people towards right.

If B does not have element to its right .no they are not in decreasing order so largest index is a right most element which does not have any ANSV anybody to the left would have found the ANSV. How long does it all take?

(Refer Slide Time: 50:34)

Prefix Minima

- For each element $A[i]$ in a vector
 - find the minimum of all $A[j]$, $j < i$
- Algorithm:
 - Find ANSV $O(n^2)$ work
 $O(1)$ time
 - For each $A[i]$
 - For each $A[j]$, $j < i$
 - $\text{PrefixMin}[i] = A[j]$ where j is the largest index such that $A[j]$ has no smaller value on its left

Diagram: A sequence of elements in circles: $m+$, $m+$, $m+$, $m+$, m , b , a . A red circle highlights the second $m+$. A red arrow points from the second $m+$ to the b .

We have just done that as routine right exactly me. I going to ask everybody who of you has an ANSV some of them will have it some of them would not have ones and zeroes I want to find the right most one which has no ANSV. So if you do not have and I am going to find the right most one element suppose it is sorted in deceasing order for everybody to the left is greater no nobody less on your left.

Some of them has some of them do not same some of them are one some of them are o. You need find the nearest one. some of them are one. so now we are going to take a break.