DOWNLOAD PDF BASIC AND STRUCTURAL DATA

Chapter 1 : Introduction to Basic and Advanced Data Structures and Algorithms

This chapter explains the basic terms related to data structure. Data type is a way to classify various types of data such as integer, string, etc. which determines the values that can be used with the corresponding type of data, the type of operations that can be performed on the corresponding type.

Published online Nov Bluhm ,1 Cole H. Goodsell ,3 John D. Berman ,2 Philip E. Bourne ,1,4 and Stephen K. This article has been cited by other articles in PMC. Our efforts over the past 2 years focused on enabling a deeper understanding of structural biology and providing new structural views of biology that support both basic and applied research and education. Herein, we describe recently introduced data annotations including integration with external biological resources, such as gene and drug databases, new visualization tools and improved support for the mobile web. We also describe access to data files, web services and open access software components to enable software developers to more effectively mine the PDB archive and related annotations. Our efforts are aimed at expanding the role of 3D structure in understanding biology and medicine. The PDB archive is the singular global repository of the 3D atomic coordinates and related experimental data of proteins, nucleic acids and complex assemblies. The PDB was one of the first open access digital resources since its inception with only seven structures in 3 , 4. Together, the four wwPDB partners develop common deposition and annotation services 10 , define data standards and validation criteria in collaboration with the user community 11 and task forces 12 â€” 15 , develop data dictionaries 16 , 17 and curate data depositions according to agreed standards 18 , On the technical side, we report improvements to visualization, mobile support, internal software development processes, programmatic access to PDB data and annotations using web services and access to software libraries. Finally, we describe expansion of our educational offerings, PDB http: Our tools and resources enable scientists to discover new relationships between sequence, structure and function, gain new insights and create new biological or biochemical hypotheses using atomic level information. Representation of structures in the context of biology and medicine and related educational resources are internet-accessible tools for high school, undergraduate and graduate level courses, and more recently Massive Open Online Courses. For X-ray structures, however, only the atomic coordinates of the asymmetric unit representing the smallest portion of a crystal structure to which symmetry operations can be applied to generate the complete unit cell are deposited to the PDB. The asymmetric unit is, in many cases, not the biologically relevant form of a multimeric complex. One and occasionally multiple biological assemblies are assigned to each PDB entry based on experimental evidence or prediction of the most likely biological assembly by the program PISA We characterize the stoichiometry and symmetry of biological assemblies and provide query and visualization tools to find and analyze them. A large fraction of protein complexes are symmetric. To systematically characterize symmetry, pseudo-symmetry and protein stoichiometry subunit composition across all biological assemblies in the PDB archive, we have developed an efficient algorithm with which to characterize symmetry, extending earlier work by Levy Then, the centroids of identical or homologous subunits are superposed to generate an initial transformation matrix. Point group and symmetry axes for the complex are then determined from the transformation matrices. Helical symmetry is determined in a similar manner. Only protein chains with at least 20 residues are considered in these calculations. Based on the sequence clustering of the biological assemblies, the PDB currently contains about 48 monomers, 38 homomers and 12 heteromers. Queries for stoichiometry and symmetry are available in the Advanced Search interface. Symmetry is illustrated via color-coding and inclusion of an appropriately shaped polyhedron enclosing the multimeric complex.

Chapter 2 : List of data structures - Wikipedia

Data structure introduction refers to a scheme for organizing data, or in other words a data structure is an arrangement of data in computer's memory in such a way that it could make the data quickly available to the processor for required calculations.

Usage[edit] Data structures serve as the basis for abstract data types ADT. The data structure implements the physical form of the data type. For example, relational databases commonly use B-tree indexes for data retrieval, [6] while compiler implementations usually use hash tables to look up identifiers. Usually, efficient data structures are key to designing efficient algorithms. Some formal design methods and programming languages emphasize data structures, rather than algorithms, as the key organizing factor in software design. Data structures can be used to organize the storage and retrieval of information stored in both main memory and secondary memory. Thus, the array and record data structures are based on computing the addresses of data items with arithmetic operations , while the linked data structures are based on storing addresses of data items within the structure itself. Many data structures use both principles, sometimes combined in non-trivial ways as in XOR linking. The efficiency of a data structure cannot be analyzed separately from those operations. This observation motivates the theoretical concept of an abstract data type , a data structure that is defined indirectly by the operations that may be performed on it, and the mathematical properties of those operations including their space and time cost. List of data structures There are numerous types of data structures, generally built upon simpler primitive data types: Elements are accessed using an integer index to specify which element is required. Typical implementations allocate contiguous memory words for the elements of arrays but this is not always a necessity. Arrays may be fixed-length or resizable. A linked list also just called list is a linear collection of data elements of any type, called nodes, where each node has itself a value, and points to the next node in the linked list. The principal advantage of a linked list over an array, is that values can always be efficiently inserted and removed without relocating the rest of the list. Certain other operations, such as random access to a certain element, are however slower on lists than on arrays. A record also called tuple or struct is an aggregate data structure. A record is a value that contains other values, typically in fixed number and sequence and typically indexed by names. The elements of records are usually called fields or members. A union is a data structure that specifies which of a number of permitted primitive types may be stored in its instances, e. Contrast with a record , which could be defined to contain a float and an integer; whereas in a union, there is only one value at a time. Enough space is allocated to contain the widest member datatype. A tagged union also called variant , variant record, discriminated union, or disjoint union contains an additional field indicating its current type, for enhanced type safety. An object is a data structure that contains data fields, like a record does, as well as various methods which operate on the data contents. An object is an in-memory instance of a class from a taxonomy. In the context of object-oriented programming , records are known as plain old data structures to distinguish them from objects. Language support[edit] Most assembly languages and some low-level languages, such as BCPL Basic Combined Programming Language , lack built-in support for data structures. On the other hand, many high-level programming languages and some higher-level assembly languages, such as MASM , have special syntax or other built-in support for certain data structures, such as records and arrays. For example, the C a direct descendant of BCPL and Pascal languages support structs and records, respectively, in addition to vectors one-dimensional arrays and multi-dimensional arrays. Modern languages usually come with standard libraries that implement the most common data structures. Modern languages also generally support modular programming , the separation between the interface of a library module and its implementation. Some provide opaque data types that allow clients to hide implementation details. Many known data structures have concurrent versions which allow multiple computing threads to access a single concrete instance of a data structure simultaneously.

Chapter 3 : Structures (Visual Basic) | Microsoft Docs

Learn the fundamental elements of databases and how they are structured. One common data structure is a database table, which uses records and fields to organize data.

Geodatabases are databases stored in Microsoft Access for the "personal" geodatabase , as a special collection of files for the "file-based" geodatabase , or higher-end applications e. A geodatabase stores all features and related tables, as well as other files, within a single or distributed database format. There are several advantages to using geodatabases rather than other storage formats: At ArcGIS, it is possible to create geoprocessing models for complex analyses, as well as toolboxes containing custom tools, and have these stored in a geodatabase. In terms of the basic representation of spatial features points, lines, and polygons , and their spatial referencing, the geodatabase functions the same as the other formats. To find out more about geodatabases, read the ArcGIS help topics. ESRI also publishes a textbook on data modeling that is focused around the geodatabase. Because of the simple data and file structure, shapefiles draw very quickly in ArcGIS Shapefile data files can also be managed using operating system tools, such as the Windows Explorer. The shapefile standard is public, so any software can be made to read or write shapefiles. A single shapefile represents features that are either point, line, or polygon in spatial data type. If you create a shapefile, you need to choose what feature type you want at the time of creation. Spatial referencing of shapefiles is enforced by maintaining explicit X and Y coordinates for each point or vertex in the layer. Typically, this is done at the time of data creation, where a new dataset is drawn in reference to existing datasets that are already georeferenced. For each shapefile there exist at least 3 files, the shape data stored in the. For each feature within the shapefile, there is an associated record within the attribute table. ArcInfo Coverages are a basic implementation of the vector arc-node topological model as shown in the cartoon-like images above. Like shapefiles, coverages also have associated database tables, with a one-to-one feature-to-record relationship. As you can see in the schematic for the ArcInfo polygon coverage, the coverage can be a multi-feature, or "polymorphic" dataset, composed of polygons, arcs, nodes, label points, and tics. This is due to the original way features were modeled in some of the first GIS software applications. It is a complex and sometimes unwieldy way to store data, but because many systems still use ArcInfo as the main GIS software, the coverage will be around for quite some time. Because of the complex structure of some coverage data, coverages can take a long time to draw. Also, due to intricacies of the file system storage model, coverages cannot be managed fully i. Operating system tools for file management e. For these reasons, ArcInfo coverages are often used as sources in ArcGIS, but they are frequently converted to the shapefile format for other uses. Spatial referencing of ArcInfo coverages is enforced by tics , and all other features within the coverage are spatially referenced relative to the tics. Most of the datasets we will be using throughout the term are ArcInfo coverages. Within ArcGIS, there are only minor differences between the functionality of the shapefile and the coverage. Some of these problems are avoidable, while some are not. For polygon features, shapefiles and coverages have very different spatial data structures. The greatest differences can be shown comparing how polygons are stored. While coverages use the standard arc-node topology data structure, in which adjacent polygons share common bounding arcs, shapefiles store each feature as a separate object. Here is a close up view of a few adjacent coverage polygons. When one bounding arc is moved, both polygons are affected. Figure 8 Compare this to a shapefile, in which different polygons can be moved without affecting adjacent polygons: Figure 9 The geodatabase can function in either of the ways shown above. While polygonal features in a geodatabase are stored as individual "rings" as in Figure 8 , it is possible to define topological relationships so that adjacent features are not able to be edited independently as in Figure 9. Point layers can be created from files containing single records for individual points. The source files which single records for individual points, where each record contains X and Y coordinates, as well as any other optional attribute data. In this example, there is a different record for each point representing a populated place in the dataset. The coordinates can be represented as points on a map. Vector data scale dependency For all vector datasets, you should always consider the scale dependency of spatial data. When should an airport be represented as a point,

and when should it be a polygon? If you are measuring the distance from major cities to their airports, then the cities and airports would be best represented as points. However, if you are planning wetland mitigation for an addition to an airport, then the airport boundary would be better represented as a polygon. Raster datasets are composed of rectangular arrays of regularly spaced square grid cells. Each cell has a value, representing a property or attribute of interest. While any type of geographic data can be stored in raster format, raster datasets are especially suited to the representation of continuous, rather than discrete, data. Some examples of continuous data are: Generally, cells are assigned a single numeric value, but with GRID a proprietary ArcInfo data format layers, cell values can also contain additional text and numeric attributes. In the above diagram, each feature type on the landscape buildings, elevation, roads, vegetation is represented in its own raster layer. Note that each raster layer has cells with numbers. For the buildings layer, all cell values are 2 in this case, 2 is a code for houses; other buildings would be encoded with a different value. For the elevation layer, the cell value is the elevation at the center of the cell. For roads, a value of 3 indicates a road other road features, e. For vegetation, trees have a value of 1. In this example, grass is treated as a background value and has no data value although it could have been given a different numeric value. All raster datasets are spatially referenced by a very simple method: This image shows the upper-left corner as the grid origin, with arrows representing the X and Y location of the cells. Different raster file formats may have an origin located at the lower left rather than at the upper left. Each cell or pixel contains a value representing some numerical phenomenon, or a code use for referencing to a non-numerical value. This speeds up processing time immensely in comparison to certain types of vector data processing. However, the file sizes of raster datasets can be very large in comparison to vector datasets representing the same phenomenon for the same spatial area. Also, there is a geometric relationship between raster resolution and file size. A raster dataset with cells half as large e. The following image shows the difference in cell sizes, area, and number of cells for two configurations of the same total area: Cells may either have a value 0-infinity or no value null, or no data. The difference between these is important. Null values mean that data either fall outside the study area boundary, or that data were either not collected or not available for those cells. In general, when null cells are used in analysis, the output value at a the same cell location is also a null value. Grid datasets can store either integer or floating-point decimal data values, though some other data formats can only store integer values. Typical simple image data will have strict limits on the number of unique cell values typically All raster datasets are stored in similar formats. You will want to know the difference between a pixel and a cell, even though they are functionally equivalent. A pixel short for PICture ELement represents the smallest resolvable "piece" of a scanned image, whereas a cell represents a user-defined area representing a phenomenon. A pixel is always a cell, but a cell is not always a pixel. There are many types of raster data you may be familiar with: Here are few graphical examples of raster data. Note that each image, when zoomed in, shows the same pixellated structure. A simple scanned color photographic image GIF format. Some 8-bit digital orthophotos in BIL, "band interleave by line" format. Three differently scaled views of an ArcInfo format elevation grid, showing cell outlines and elevation values. Most of the raster datasets we will use are single-band, which means that they contain a single "layer" of data. The data can represent elevation, slope, or reflectance in the case of the black-and-white digital orthophotos we will see later. These single-band images are viewed with a color mapping, so that the cell value is associated with a particular color. For the orthophotos, the color map is a value greyscale ramp. Other raster data, such as elevation models, can be mapped to color ramps that display elevation ranges, as shown in the image directly above. Multi-band raster data such as RGB images or satellite images are generally displayed with a mixture of red, green, and blue values for each different band in the image. As you can see, when any of the raster layers are displayed at larger scales, the individual cells become visible. As scale of display increases, precision also decreases, and shapes cannot be precisely represented. All spatial datasets are generalized; however, raster datasets more clearly show their level of generalization. A more complete discussion of generalization in relation to scale is addressed in Scale Issues. Here are the first few lines in the Eatonville, WA 7. The subsequent lines list elevations for the lattice mesh points cell centers.

Chapter 4 : Basic of Data Structure

Understanding Basic Data Types in R. To make the best of the R language, you'll need a strong understanding of the basic data types and data structures and how to operate on those.

Data Structure Consider the following three examples. What do they all have in common? Refrigerate about 20 minutes, stirring occasionally until mixture mounds slightly when dropped from a spoon. Data Structure A data structure is a way of organizing data that considers not only the items stored, but also their relationship to each other. Advance knowledge about the relationship between data items allows designing of efficient algorithms for the manipulation of data. Data Structure The malloc function is used to allocate memory and has the following prototype: Data Structure Record and record structures: The members of a record can be variables or other records. However, a record can not contain circular record, i. Data Structure Both the malloc and the calloc functions are used to allocate dynamic memory. Each operates slightly different from the other. Data Structure Following steps must be followed to plan any algorithm: Creating an algorithm is an art in which may never be fully automated. When we get the problem, we should first analyse the given problem clearly and then write down some steps on the paper. Data Structure The free subroutine frees a block of memory previously allocated by the malloc subroutine. Undefined results occur if the Pointer parameter is not a valid pointer. If the Pointer parameter is a null value, no action will occur. The function realloc ptr,n uses two arguments. The second argument n specifies the new size. The size may be increased or decreased.

Chapter 5 : Basic Concepts of Data Structure | Data Structure | www.nxgvision.com

Basic Concepts of Data Structure Data Structure is a way of collecting and organising data in such a way that we can perform operations on these data in an effective way. Data Structures is about rendering data elements in terms of some relationship, for better organization and storage.

References Introduction to Basic Data Structures and Algorithms Before introducing data structures we should understand that computers do store, retrieve, and process a large amount of data. A data structure should be seen as a logical concept that must address two fundamental concerns. First, how the data will be stored, and second, what operations will be performed on it? As data structure is a scheme for data organization so the functional definition of a data structure should be independent of its implementation. The functional definition of a data structure is known as ADT Abstract Data Type which is independent of implementation. The implementation part is left on developers who decide which technology better suits to their project needs. For example, a stack ADT is a structure which supports operations such as push and pop. A stack can be implemented in a number of ways, for example using an array or using a linked list. Along with data structures introduction, in real life, problem solving is done with help of data structures and algorithms. An algorithm is a step by step process to solve a problem. In programming, algorithms are implemented in form of methods or functions or routines. To get a problem solved we not only want algorithm but also an efficient algorithm. One criteria of efficiency is time taken by the algorithm, another could be the memory it takes at run time. Sometimes, we can have more than one algorithm for the same problem to process a data structure, and we have to choose the best one among available algorithms. This is done by algorithm analysis. The best algorithm is the one which has a fine balance between time taken and memory consumption. But, as we know the best exists rarely, and we generally give more priority to the time taken by the algorithm rather than the memory it consumes. Also, as memory is getting cheaper and computers have more memory today than previous time, therefore, run time analysis becomes more significant than memory. The analysis of algorithms is an entirely separate topic and we will discuss that separately. Classification Data Structures Data structures can be broadly classified in two categories - linear structures and hierarchical structures. Arrays, linked lists, stacks, and queues are linear structures, while trees, graphs, heaps etc. Every data structure has its own strengths, and weaknesses. Also, every data structure specially suits to specific problem types depending upon the operations performed and the data organization. For example, an array is suitable for read operations. Following is a quick introduction to important data structures. But modern programming languages, for example, Java implements arrays as objects and give the programmer a way to alter the size of them at run time. Arrays are the most common data structure used to store data. Arrays are unarguably easier data structures to use and access. But inserting an item to an array and deleting it from the array are situation dependent. If you want to insert an item at a particular position which is already occupied by some element then you have to shift all items one position right from the position new element has to be inserted then insert the new item. The time taken by insert operation is depend on how big the array is, and at which position the item is being inserted. The same is true about deletion of an item. If the array is unsorted then search operation is also proved costly and takes O N time in worst case, where N is size of the array. But if the array is sorted then search performance is improved magically and takes O logN time in worst case. Linked List Linked list data structure provides better memory management than arrays. Because linked list is allocated memory at run time, so, there is no waste of memory. Performance wise linked list is slower than array because there is no direct access to linked list elements. Linked list is proved to be a useful data structure when the number of elements to be stored is not known ahead of time. There are many flavors of linked list you will see: Stack Stack is a last-in-first-out strategy data structure; this means that the element stored in last will be removed first. Stack has specific but very useful applications; some of them are as follows: Solving Recursion - recursive calls are placed onto a stack, and removed from there once they are processed.

Chapter 6 : Introduction to Data Structures and Algorithms | Studytonight

DOWNLOAD PDF BASIC AND STRUCTURAL DATA

Introduction to the Basic Data Structure Challenges Data can be stored and accessed in many different ways, both in Javascript and other languages. This section will teach you how to manipulate arrays, as well as access and copy the information within them.

Chapter 7 : Using Drillthrough on Structure Data (Basic Data Mining Tutorial) | Microsoft Docs

Data Structures are the programmatic way of storing data so that data can be used efficiently. Almost every enterprise application uses various types of data structures in one or the other way. This tutorial will give you a great understanding on Data Structures needed to understand the complexity of enterprise level applications and need of.

Chapter 8 : Data structure - Wikipedia

Structural equation modeling (SEM) is a methodology for representing, estimating, and testing a network of relationships between variables (measured variables and latent constructs). This tutorial provides an introduction to SEM including comparisons between.

Chapter 9 : The GIS Spatial Data Model

Our + Data Structure questions and answers focuses on all areas of Data Structure subject covering + topics in Data Structure. These topics are chosen from a collection of most authoritative and best reference books on Data Structure. One should spend 1 hour daily for months to learn and.