AS Computer Science
Algorithms and programs (5)

Key terms

Term	Definition
Compression	The process of reducing file size. It allows more data to be stored on disk and enables files to be transferred more quickly.
Testing strategy	A strategy to ensure that a final solution to a problem meets the original requirements of the user.
Test plan	A document that details the scope, approach, resources, and schedule of intended test activities. Used to record the test data used and test outcomes.

Compression algorithms

Lossless compression

All data that was in the original file remains after the file is uncompressed. All of the information is completely restored. Generally used for text or spreadsheet files, where words or financial data must not be lost.

The efficiency of a lossless algorithm can be assessed according to the amount of shrinkage of the source file, expressed as a ratio.

$$compression\ ratio = \frac{size\ after\ compression}{size\ before\ compression}$$

OR as a percentage

$$saving\ \% = \frac{size\ before\ compression - size\ after\ compression}{size\ before\ compression}\ \%$$

EXAMPLE

Image source file of 256 x 256 bytes = 65536 bytes, compressed to 16384 bytes: -
16384 / 65536 = ratio of 1 : 4 and
a saving percentage = 75 %

Lossy compression

Reduces a file by permanently eliminating certain information. When the file is uncompressed, only a part of the original information is still there. Generally used for video and audio files, where a certain amount of information loss will not be detected by most users.

In compression of audio and video data, the final arbiter of quality, fidelity, or distortion is subjective and determined by the user.

EXAMPLE

A photograph of a blue sky would capture many variations in shades of blue. These can be replaced to use only the most frequently occurring shades and reference a much reduced set of colours.

Software Testing

Involves checking whether actual results match expected results and that the software system is defect free. Testing is important because software bugs could be expensive to rectify or even dangerous to end users.

Software testing aims to ensure:

- cost-effectiveness
- data security
- product quality
- customer satisfaction.

Testing may be sub divided into three categories

Functional testing: testing each function of the software application, by providing appropriate input and verifying the output against the functional requirements.

Performance testing: testing the speed, response time, reliability, resource usage, scalability of a software program under expected workload.

Regression testing: testing to confirm that program updates or that code changes have not adversely affected existing features.

Software Testing strategies

Unit testing: testing a section or unit of code in isolation to verify its correctness. A unit may be an individual function, method, procedure, or module.

Integration Testing: testing to expose defects in the interaction between software units when they are integrated.

System testing: testing of compiled software as a whole to check overall functionality, security etc.

Acceptance Testing: formal testing conducted to determine whether a system satisfies its acceptance criteria and to enable the customer to determine whether to accept the system.

Alpha Testing: a type of acceptance testing performed to identify bugs before releasing the product to real users or the public.

Beta Testing: final testing before releasing application for commercial purpose, typically done by end-users.

Whitebox testing: testing by the developers based on the inner workings of an application.

Blackbox testing involves testing from an external or end-user perspective.

Test data

The design of test data is crucial to the success of the testing process. Test data should include:

Typical/standard data

Data that is valid and should be processed correctly by the program, used to check validation rules and to confirm accuracy by comparing results with previously calculated outcomes, i.e. testing for logical errors.

Extreme/boundary data

Data that is valid but is at the extreme boundary of an acceptable range of values. Used to confirm data ranges have been correctly set.

Erroneous data

Data that would cause the program to fail if not validated and rejected.