Project Name- Decoding many strings stored in an array from Base64 efficiently

Project Category- Encoding/Decoding, Base64

Pre-requisites- Base64 Decoding, ASCII Values

Motivation-

Earlier, I have made a small project to show how to encode many strings efficiently in Base64 in C++. The project can be found at – (https://github.com/gbelwariar/Self-Made-Projects/tree/master/Encode-Base64)

This project is about how to "get back" the original string , i.e.- how to decode a string from Base64.

Inbuilt functions in various languages to Encode/Decode strings -

Language	Encode	Decode	Notes
PHP	`base64_encode($string);`	`base64_decode($string);`	
Perl	`encode_base64($string);`	`decode_base64($string);`	Requires `use MIME::Base64;`
C#	`System.Convert.ToBase64String(System.Text.Encoding.UTF8.GetBytes(plainTextBytes));`	`System.Text.Encoding.UTF8.GetString(System.Convert.FromBase64String(base64EncodedData));`	
Java	`Base64.encodeBase64(string);`	`Base64.decodeBase64(string);`	Requires `import org.apache.commons.codec.binary.Base64;`
JavaScript	`btoa(string);`	`atob(string);`	<= IE9 is unsupported
Ruby	Base64.encode64('string')	Base64.decode64(enc)	Requires `require "base64"`

Explanation / Algorithm-

The below text has been taken from - http://www.hcidata.info/base64.htm . I felt that the explanation given in this website is the best I can find on the web in terms of simplicity.

Suppose we have a Base64 Encoded String - **TWFyeSBoYWQ=**

Decoding "TWFyeSBoYWQ=" from Base 64 to ASCII

The first thing to note is the '=' at the end of the Base 64 encoded string. A Base 64 encoded string will have zero, one or two '='s at the end. As '=' is not part of the Base 64 encoding, it can only ever appear at the end and has a special meaning.

The stages to convert the Base 64 encoded data to ASCII is:

1. Take each letter and find its position offset (0-63) within ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/ to give us 11 decimal numbers
2. These decimal numbers are converted to bit strings, each with 6 bits
3. As there was a '=' at the end of the Base 64 encoded data, remove 2 '0's from the end of the bit string. Had there been two '='s at the end of the Base 64 encoded data, 4 '0's would have been removed from the end of the bit string.
4. Split the bits into groups of 8
5. Use each group of 8 bits to be the ASCII code for an ASCII character or ASCII control code

Implementation-

Since the main objective of this program is to quickly/efficiently encode the strings stored in an array, so we are using two hash maps one to "*Base64 Character to corresponding 6-Bit Representation*" and another one to map from "*8-Bit Representation to ASCII Character*"

We will hence use two **unordered_map** to quickly insert and retrieve the values.

Time Complexity-

We insert and find the elements in the **unordered_map** in O(1) Amortized time.
Hence the overall time complexity of this program is **O(No_of_Strings * Avg_length_of_each_String)**

Auxiliary Space / Space Complexity-

We are using two **unordered_map** containing 128 and 64 elements . So considering them as constant space, we can say that we have used **O (max(length_of_strings_in_array))** auxiliary space as we are storing the cumulative bits in the string-'*grouped*'.

References-

http://www.hcidata.info/base64.htm