

O NAUCZANIU ALGORYTMÓW W XXI WIEKU – STUDIUM PRZYPADKU

*Mariusz Kaniecki
Wydział Matematyki i Informatyki
Uniwersytet Mikołaja Kopernika w Toruniu
e-mail kanies@mat.umk.pl*

Abstract. *The purpose of this paper is to examine the implementation of the project-based learning approach in teaching the course Algorithms and Data Structures II within the computer science degree program.*

1. Wstęp

Niniejsza praca ma na celu przedstawienie wykorzystania metody projektowej w nauczaniu przedmiotu Algorytmy i struktury danych II w ramach studiów na kierunku informatyka realizowanych na Wydziale Matematyki i Informatyki UMK.

Nauka algorytmów nie jest zadaniem banalnym. Jako opcję minimum można traktować spełnienie przez studenta wymagań w zakresie wiedzy, poprzez poznanie zadanych algorytmów. Ich dokładne zrozumienie wraz z umiejętnością programowania powinno umożliwić implementację poznanych algorytmów. Takie zadania, choć z pewnością cenne, mogą jawić się studentom jako nieatrakcyjne. Wszak bogate biblioteki współczesnych języków programowania zawierają implementacje o wiele większej liczby algorytmów niż te poznane w ramach pojedynczego przedmiotu. Duże modele językowe w kilka sekund wskażą algorytmy z bazy poznanej przez studenta. Wydaje się zatem koniecznością, stawianie nie tylko na prostą implementację, ale również modyfikację, a także analizę i omówienie przez studentów zaproponowanych rozwiązań.

W ramach zaliczenia przedmiotu Algorytmy i struktury danych II studenci mają stworzone warunki do zespołowej pracy nad projektem programistycznym opartym na poznanych algorytmach i strukturach danych. W tej pracy staramy się przedstawić sposób prowadzenia takich zajęć.

2. Formalne informacje o przedmiocie

Algorytmy i struktury danych II od roku akademickiego 2023/2024 są przedmiotem obowiązkowym dla studentów 2. roku informatyki (zarówno studiów inżynierskich, jak i licencjackich). Wcześniej przez dwa lata studenci tych studiów mogli w ramach

przedmiotów do wyboru zaliczyć zajęcia Algorytmy II, które były niejako próbą generalną przedmiotu, który miał stać się obowiązkowy. W ramach obu tych kursów realizowane jest 30 godzin wykładu, którym towarzyszy 30 godzin zajęć laboratoryjnych. Krótki opis przedmiotu umieszczony w systemie USOS przedstawia ten przedmiot w następujący sposób [2]:

„Przedmiot jest kontynuacją przedmiotu Algorytmy i struktury danych. Jego celem jest przedstawienie bardziej zaawansowanych algorytmów i ich zastosowań oraz struktur danych używanych w implementacjach tych algorytmów. Analiza algorytmów i struktur danych dotyczy złożoności obliczeniowej implementacji algorytmów.”

W tym samym miejscu określono również następujące efekty kształcenia:

1. W zakresie wiedzy studenta:
 - a. zna zaawansowane struktury danych, m.in. dla uporządkowanego multizbioru, drzewa licznikowe, potęgowe (Fenwicka), drzewa przedziałowe, kolejka minimów.
 - b. zna podstawowe pojęcia teorii sieci oraz podstawowe algorytmy wyznaczające maksymalny przepływ w sieci.
 - c. zna podstawowe algorytmy geometryczne, m.in. położenie punktu względem odcinka, przynależność punktu do wielokąta, znajdowanie otoczki wypukłej, metoda zamykania.
 - d. zna podstawowe algorytmy tekstowe, m.in. algorytmy wyszukiwania wzorca, algorytm Rabina-Karpa, algorytm Huffmana, algorytm Manachera.
 - e. zna algorytmy aproksymacyjne wyznaczające: pokrycie wierzchołkowe, pokrycie zbioru, rozwiązujący problem sumy podzbioru.
 - f. zna przykłady zastosowań omawianych algorytmów.
2. W zakresie umiejętności studenta:
 - a. umie zastosować poznane algorytmy na przykładach.
 - b. potrafi samodzielnie wyszukać (w internecie lub literaturze) oraz zastosować na odpowiednich przykładach algorytmy rozwiązujące zadane problemy.
 - c. potrafi zaimplementować poznane algorytmy.
 - d. potrafi pracować indywidualnie i w zespole informatyków, w tym także potrafi zarządzać swoim czasem oraz podejmować zobowiązania i dotrzymywać terminów.
3. W zakresie kompetencji społecznych:
 - a. przekazuje innym swoją wiedzę i przemyślenia w zrozumiały sposób; właściwie rozumie sformułowania pytań i problemów, poprawnie posługuje się terminologią fachową.
 - b. rozumie potrzebę ciągłego uczenia się.

Jak widzimy, w ramach kursu studenci mają otrzymać zarówno sporą porcję wiedzy przez poznanie nowych (w stosunku do wcześniejszych przedmiotów) algorytmów i struktur danych, jak i ulepszyć umiejętności praktycznego użycia poznanych

algorytmów w szczególności przez pracę w zespole. Do osiągnięcia tego celu należy stworzyć studentom warunki do pracy w takiej formie. W szczególności do dzielenia się otrzymanymi zadaniami w taki sposób, aby każdy z programistów był w stanie dotrzymać wyznaczonych terminów.

2.1. Charakterystyka studenta

Przedstawimy teraz krótką charakterystykę studenta na 4. semestrze studiów informatycznych, przystępującego do zajęć z Algorytmów i Struktur danych. Skupimy się jedynie na przedmiotach i umiejętnościach wpływających w istotny sposób na omawiany przedmiot.

Podstawowe pojęcia związane z algorytmami student poznaje na pierwszym semestrze w ramach przedmiotu Podstawy algorytmiki i programowania (patrz [4]). Kurs ten ma wymiar 30 godzin wykładu i 45 godzin zajęć laboratoryjnych, w ramach których studenci poznają pojęcie algorytmu oraz jego specyfikacji, formy prezentacji algorytmu (schemat blokowy, pseudokod, język programowania). Omawiane są również pojęcia związane ze złożonością obliczeniową - funkcja kosztu, funkcja pesymistycznej czasowej złożoności algorytmu, rząd wielkości, notacja asymptotyczna. Studenci poznają podstawowe metody projektowania algorytmów - zachłanną, „dziel i zwyciężaj”, programowanie dynamiczne, a także pojęcie rekurencji. Przedstawiane jest im również przegląd rozwiązań wybranych problemów obliczeniowych, w szczególności algorytmy wyszukiwujące oraz algorytmy sortowania. Uzupełnienie wiedzy algorytmicznej stanowi fragment kursu poświęcony wskaźnikom oraz dynamicznym strukturom danych – listom, kolejkom, stosom. W ramach tego przedmiotu studenci poznają podstawy języka C++, jednak bez wykorzystania jego obiektowości.

Wiedza i umiejętności zdobyte w ramach przedmiotu Podstawy algorytmiki i programowania pogłębiane są na Algorytmice i programowanie w języku Python (patrz [4]), w ramach którego przewidziane jest 30 godzin laboratorium, wykład do tego przedmiotu nie jest prowadzony. Jak czytamy w opisie przedmiotu: „doskonali myślenie algorytmiczne i logiczne wnioskowanie. Głównym celem jest przygotowanie do rozwiązywania problemów i programowania podstawowych algorytmów, struktur danych w języku programowania Python.” Studenci oprócz pogłębienia wiedzy algorytmicznej, poznają kolejny język programowania, jakim jest Python. Nie da się ukryć, że bogactwo bibliotek tego języka, zawierających implementacje licznych algorytmów w istotny sposób wpływa na postrzeganie przez studentów zadań polegających na prostej implementacji algorytmów.

Zajęciami bezpośrednio poprzedzającymi Algorytmy i struktur danych II jest podstawowy kurs 30 godzin wykładu i 30 godzin zajęć laboratoryjnych z Algorytmów i struktur danych (patrz [1]). Studenci w jego ramach dostają pogłębioną szkołę formalizmu związanego z algorytmami, zarówno z ich złożonością (pełna funkcja kosztu, koszt zamortyzowany, złożoność asymptotyczna, notacje "O", "Ω", "Θ", metody wyznaczania funkcji

kosztu algorytmów rekurencyjnych), jak i z formalnym dowodzeniem poprawności – częściowej (systemy formalne Hilberta i Hoare'a, niezmiennik pętli) i całkowitej (własność stopu, zbieżność pętli). Poznają również drzewiaste struktury danych - drzewo przeszukiwań binarnych, kopiec binarny. Omawiane są elementy teorii grafów oraz sposoby reprezentacji grafów w komputerze. Studenci poznają oraz implementują algorytmy grafowe, np. przeszukiwanie grafów w głąb (DFS) i wszerz (BFS), znajdowanie najkrótszej drogi w grafach z wagami (algorytmy Dijkstry i Bellmana-Forda), algorytmy Kruskala i Prima znajdowania minimalnego drzewa rozpinającego.

Student informatyki na Wydziale Matematyki i Informatyki UMK w Toruniu, rozpoczynając 4. semestr studiów, zna kilka języków programowania. Pierwszym językiem programowania jest C++, który jest podstawowym językiem w ramach zajęć z Podstawy algorytmiki i programowania. W czasie trzeciego semestru student poznaje bardziej zaawansowane składniki języka C++ w ramach przedmiotu Programowanie II. W szczególności omawiany jest przede wszystkim paradygmat programowania obiektowego w tym języku. Język C studenci poznają w czasie zajęć z Programowania I oraz Programowania sieciowego. Trzecim językiem programowania jest wspomniany wyżej Python, z którym studenci zapoznają się w ramach przedmiotu Algorytmika i programowanie w języku Python. Równolegle z zajęciami z Algorytmów i struktur danych II, studenci uczestniczą w przedmiocie Programowanie III(Java) lub Programowanie III (C#). Na zajęciach w wymiarze 45 godzin laboratoryjnych studenci poznają język Java lub C#, w zależności od wybranej grupy.

Ostatnim przedmiotem, który warto by w tym miejscu omówić, jest Inżynieria oprogramowania (patrz [4]). Przedmiot jest realizowany na trzecim semestrze studiów w wymiarze 15 godzin wykładu oraz 15 godzin laboratoriów. W czasie tych zajęć studenci zdobywają wiedzę związaną z planowaniem i zarządzaniem projektem programistycznym, a także jego testowaniem. W szczególności poznają systemy kontroli wersji, klasyczne i zwinne metodyki tworzenia oprogramowania, wzorce projektowe.

Podsumowując student rozpoczynający kurs omawiany w niniejszej pracy ma wiedzę dotyczącą podstawowych algorytmów od strony zarówno formalnej, jak i praktycznej. Wie czym jest złożoność obliczeniowa i w jaki sposób można ją szacować. Zna podstawy teorii grafów oraz wybrane algorytmy dla problemów grafowych. Używa struktur danych i umie oszacować złożoność poszczególnych operacji na tych strukturach. Ponadto jest wyposażony w umiejętność programowania w przynajmniej kilku językach. Potrafi programować zarówno w językach strukturalnych, jak i obiektowych. Poznał sposoby i narzędzia projektowania i tworzenia oprogramowania.

3. Praca na zajęciach oraz sposób zaliczenia przedmiotu

Omówimy teraz sposób realizacji poszczególnych składowych kursu.

3.1. Wykład

Wykład jako podająca metoda dydaktyczna w większości ma formę wykładu konwencjonalnego. W ramach wykładu przedstawiane są algorytmy wraz z przykładowymi przebiegami algorytmów. Część wykładów poświęcona jest formalnym dowodom poprawności przedstawionych rozwiązań. Omawiana jest również ich złożoność obliczeniowa. Wykład można podzielić na 5. głównych składowych (patrz [2]):

1. Struktury danych - na tę część składają się: struktury dla uporządkowanego multizbioru, zrównoważone BST, drzewa licznikowe, potęgowe (Fenwicka), drzewa przedziałowe, kolejka minimów, maksimów.
2. Przepływy w sieciach w ramach której studenci poznają algorytmy znajdowania maksymalnego przepływu w sieciach (algorytm Forda-Fulkersona, Edmondsa-Karpa), przepływu z minimalnym kosztem, a także przykład zastosowania algorytmu maksymalnego przepływu do wyznaczanie maksymalnego skojarzenia w grafach dwudzielnych.
3. Algorytmy geometryczne w tej części studenci uczą się algorytmicznego określenia położenie punktu względem odcinka, przynależność punktu do wielokąta (zarówno wypukłego, jak i dowolnego), znajdowania otoczki wypukłej (algorytm Jarvisa i algorytm Grahama), testowania przecinania się odcinków.
4. Algorytmy tekstowe główną częścią tej składowej są algorytmy wyszukiwania wzorca w tekście: algorytm naiwny, Rabina-Karpa, Boyera-Moore'a i algorytm KMP, ale również haszowanie, czy kompresja z wykorzystaniem algorytmu Huffmana.
5. Algorytmy aproksymacyjne w ramach tej krótkiej części studenci poznają algorytmy wyznaczające przybliżone rozwiązania dla problemów pokrycie wierzchołkowego, pokrycie zbioru, czy problemu sumy podzbioru.

3.2. Laboratoria

Grupy laboratoryjne składają się z maksymalnie 16 studentów, zajęcia odbywają się w salach komputerowych. Aktywności w ramach tych zajęć można podzielić na trzy grupy, które zajmują zbliżoną liczbę godzin:

1. Zajęcia teoretyczne poświęcone na udowodnienie wybranych faktów przedstawionych na wykładzie bez dowodu, adaptacji algorytmów do innych problemów, czy analizie poszczególnych struktur danych.
2. Zajęcia praktyczne poświęcone implementacji wybranych algorytmów.
3. Zajęcia projektowe poświęcone podziałowi i podsumowywaniu zadań związanych z końcowym projektem.

Taki podział zajęć stwarza studentom możliwość osiągnięcia efektów kształcenia przewidzianych w programie studiów zarówno w zakresie umiejętności, jak i w zakresie kompetencji społecznych.

3.3. Zaliczenie przedmiotu

Zaliczenie przedmiotu odbywa się na podstawie przygotowanych i zaprezentowanych projektów. W ramach projektu studenci pracują w 3-4 osobowych zespołach, w które dobierają się już na jednym z pierwszych zajęć. Przez cały semestr dzielą się zadaniami w ramach projektu, którego opis dostają w pierwszym miesiącu zajęć. Co ważne opis problemu jest bardzo ogólny, pisany nieformalnym językiem, co wymusza na studentach postawienie i sformalizowanie konkretnego problemu, wygenerowanie przykładowych danych wejściowych wraz ze spodziewanym poprawnym rozwiązaniem, zaproponowanie zasad i teorii dla ogólnie postawionego problemu. Dopiero po tym wstępnym etapie przychodzi czas na wybór konkretnych algorytmów i struktur danych, następnie ich implementację i testowanie. Zauważmy, że wpisuje się to w model myślenia komputacyjnego (patrz [3]), którego algorytmika jest jedynie jednym ze składowych. Dzięki regularnym zajęciom projektowym w ramach laboratoriów prowadzący grupę ma stały wgląd w postępy pracy zespołu. Na bieżąco może wesprzeć zespół w pokonywaniu kolejnych kroków tego złożonego procesu.

W ramach projektu każda grupa ma za zadanie przygotować rozwiązanie składające się z:

- dokumentacji,
- plików programów,
- opisu poprawności rozwiązania,
- raportów przeprowadzonych testów,

a efekty pracy powinny być zapisywane z wykorzystaniem systemu kontroli wersji. Na koniec semestru każda z grup prezentuje wyniki swojej pracy przed innymi studentami i prowadzącymi. Prowadzący grupę laboratoryjną musi otrzymać dostęp do repozytorium projektu. W ten sposób wkład pracy poszczególnych członków zespołu może być względnie łatwo zweryfikowany. Dodatkowo studenci proszeni są o wypełnienie ankiety, w której opisują: harmonogram projektu, czyli czynności i kolejność ich wykonania, podproblemy wyszczególnione z projektu i algorytmy zastosowane do ich rozwiązania, podział pracy pomiędzy członkami zespołu. Ponadto wskazują podproblemy, dla których uzyskano dokładne wyniki. Są również proszeni o ocenę własnego zaangażowania w projekt.

4. Przykładowy projekt

W tym rozdziale omówimy przykładowy projekt, z którym studenci pracowali w czasie jednego z kursów. Autor ostrzega, że projekt zawiera nawiązania do twórczości

J.R.R. Tolkiena i osoby, które nie chcą poznać przedwcześnie wybranych faktów z zakończenia „Władcy Pierścieni” winny najpierw ukończyć to dzieło przed zagłębieniem się w niniejszy rozdział.

Studentom przedstawiony jest opis sytuacji problemowej. Poprzedzony informacją o tym, że należy: „odfiltrować informacje istotne od nieistotnych; sformalizować problem i podać jego specyfikację; do rozwiązania wykorzystać odpowiednie struktury danych oraz algorytmy; opisać teoretycznie rozwiązanie; przeanalizować jego optymalność oraz złożoność czasową/pamięciową; zaimplementować rozwiązanie w dowolnym języku programowania; przeprowadzić testy na różnych przykładach.”

4.1. Pola, browary i karczmy

Pierwszą część projektu stanowi opis Shire – kraju, którego dostatek zależy od ilości piwa w karczmach. Wiemy, że w kraju tym uprawia się jęczmień, pracują browary, słynie też z licznych karczm. Niestety drogi w tym miejscu mają jedynie określoną przepustowość. Treść przedstawiona studentom:

„Burmistrz Shire Samwise, chcąc zapewnić dostatek mieszkańcom swojego pięknego kraju, marzy o dobrze zaopatrzonych karczmach. W tym celu zarządził spis pól, na których uprawiany jest jęczmień, zna więc ich położenie i wie, ile średnio jęczmienia wyrasta na jednym poletku. Zna też rozmieszczenie browarów, z których każdy może przetworzyć określoną liczbę jęczmienia.

Jako że lud Shire jest pracowity i uczciwy, w każdym z browarów z tony jęczmienia uzyskuje się jednakową ilość piwa. Samwise osobiście odwiedził i zna położenie wszystkich karczm w swoim kraju. Burmistrz w swych pamiętkach posiada również mapę przygotowaną przez Bilba, gdzie oprócz zaznaczonych ulubionych ścieżek spacerowych, autor zaznaczył również ilość zboża oraz piwa, którą można przewieźć pomiędzy poszczególnymi skrzyżowaniami.”

Na tym etapie studenci mogą zauważyć, że opis pasuje do pewnego grafu ważonego, w którym wyróżnione wierzchołki mają dodatkowe cechy (mogą być polem, browarem lub karczmą) browary dodatkowo mają określone możliwości produkcyjne. To studenci muszą zdecydować, jak interpretować poszczególne informacje, na przykład, czy drogi są jedno, czy dwu kierunkowe, czy przepustowość dotyczy osobno jęczmienia, a osobno piwa (bo najpierw trzeba dostarczyć jęczmień do browarów, żeby później przewozić z nich piwo), czy ich sumy. Od tych decyzji zależy, na ile będzie można zastosować poznane algorytmy.

4.2. Tylko planowanie zapewnia dostatek

W kolejnej części projektu dowiadujemy się, że mamy do dyspozycji komputer, który musimy wykorzystać do maksymalizacji piwa dostarczanego do karczm.

„Okazało się, że pośród wielu łupów, które Bilbo przywiózł z wyprawy z krasnoludami, znalazło się urządzenie nazywane przez dużych ludzi komputerem. Samwise chce użyć go do znalezienia maksymalnej ilości trunku, jaką można dostarczyć do karczm w jego kraju (najpierw należy je wyprodukować w browarach, a do tego potrzebny jest jęczmień, który wyhodują hobbici na swoich polach). Pomóżcie Samwisowi przygotować odpowiednią strategię. Zaproponujcie jak informacje o polach, browarach, karczmach i drogach reprezentować w komputerze. Opracujcie sposób znalezienia maksymalnej ilości piwa, którą można dostarczyć do karczm w Shire.”

Problem ten jest bliski znajdowaniu maksymalnego przepływu w sieci. Należy jednak uwzględnić, że przepływ powinien przechodzić przez browary, w których z jęczmienia będzie produkowane piwo. Poprawnym (choć oczywiście nieoptymalnym rozwiązaniem) będzie dwukrotne użycie algorytmu znajdowania maksymalnego przepływu (w wersji z kilkoma źródłami i ujściami) najpierw dla sieci z pól do browarów, a następnie z browarów do karczm. Jest to przykład redukcji do prostszego problemu, które może stanowić bazę do dalszej pracy nad projektem.

4.3. Zmniejszyć koszty nie tracąc piwa

Kolejnym problemem okazuje się stan dróg w naszej krainie, który nie pozwala na realizację znalezionej planu.

„Niestety, po okresie panowania Sharkeya wiele dróg w Shire zostało zniszczonych, przez co wozy wiozące zboże i inne produkty często ulegają awarii i nie zawsze udaje się zrealizować zadania wcześniej zaplanowane. Samwise zlecił jednak Peregrinowi oraz Meriadokowi oszacowanie kosztu naprawy każdej z dróg w ich krainie. W planie transportu zboża i piwa chciałby uwzględnić te drogi, których naprawa będzie możliwie najtańsza.

Pamięta jednak, że dla hobbitów najważniejszymi sprawami są dobre jedzenie i trunk, nie chce więc zmniejszać ilości piwa dostarczanego do karczm. Zmodyfikujcie swoje rozwiązanie tak, żeby przy zachowaniu ilości przewożonego towaru koszt naprawy dróg, po których poruszają się transporty, był możliwie najmniejszy.”

W tej części najistotniejszą decyzją jest, jak rozumieć koszt naprawy dróg. Czy jego wysokość jest proporcjonalna do ilości towaru przewiezonego daną drogą, czy dla ustalonej drogi jest stały. Drugi sposób wydaje się bardziej naturalny. Jednak w przypadku pierwszego łatwiej można zastosować poznany na wykładzie algorytm przepływu z minimalnym kosztem. Tu znów wybór zależy od studentów, ich ambicji i możliwości. Jak się okazuje pracując w grupie, studenci nierzadko nie zadowolają się najprostszym rozwiązaniem, starając się je zoptymalizować.

4.4. Algorytmy geometryczne i tekstowe

Końcowa część treści projektu wymusza wykorzystanie algorytmów geometrycznych oraz tekstowych.

„Okazało się jednak, że w różnych ćwiartkach Shire ilości jęczmienia wyrastające na jednym polu różnią się od siebie. W tym celu Samwise kazał zebrać współrzędne punktów granicznych każdej ćwiartki (każda z ćwiartek okazała się być wielokątem wypukłym, rozłącznym z pozostałymi ćwiartkami). Wie też, ile jęczmienia wyrasta na polu w poszczególnych ćwiartkach. Zmodyfikujcie swoje rozwiązanie uwzględniając te nowe informacje zebrane przez burmistrza Sama.”

Chociaż w tym fragmencie wystarczy prosta implementacja algorytmu znajdowania otoczki wypukłej i sprawdzania należenia punktu do wielokąta, to dla problemów tych łatwo można generować duże zbiory danych. Studenci są zachęceni, do porównania poznanych metod przez wykonanie testów wydajnościowych, przedstawienie ich wyników graficznie i opisanie w czasie prezentacji.

W podobny sposób można traktować ostatni akapit projektu. Istotnym jest tu znajdowanie wzorca w tekście, dla tego problemu studenci poznają kilka algorytmów, które można porównywać między sobą.

„Burmistrz chce również wykorzystać skromne zasoby pamięci komputera, żeby zapisać sobie Wasze rozwiązania na przyszłość. Marzy o tym, żeby szybko wyszukiwać w tych rozwiązaniach słów: „piwo”, „jęczmień”, „browar” oraz innych, które przyjdą mu kiedyś do głowy. Chciałby przetestować kilka sposobów wyszukiwania słów. Zaproponujcie odpowiednie rozwiązania.”

Warto zwrócić uwagę na informacje o „skromnych zasobach komputera” jest to sposób na zwrócenie studentom uwagi na istotę złożoności proponowanych rozwiązań.

5. Podsumowanie

Algorytmika jest trudnym działem informatyki. Projektowanie oraz analiza algorytmów wymaga zarówno ścisłego rozumowania, jak i kreatywnego podejścia. Próba zmierzenia się z problemem projektowania algorytmów nierzadko wymaga opuszczenia strefy komfortu i podążania ścieżkami, które niekoniecznie prowadzą do poprawnego rozwiązania. Dzięki pracy w zespołach i postawieniu przed studentami problemów, których rozwiązanie wydaje się podobne, do znanych studentowi algorytmów, stwarzane są przyjazne warunki do rozwoju. Dodatkowo do kreatywnego podejścia do problemów zachęcać ma luźna interpretacja treści projektów, która umożliwi dostosowanie trudności zadania do aktualnych możliwości grupy lub studenta.

Literatura

1. Algorytmy i struktury danych USOS UMK, https://usosweb.umk.pl/kontroler.php?_action=katalog2/przedmioty/pokazPrzedmiot&prz_kod=1000-I1ASD&callback=g_31d22e06 ; ostatni dostęp 10.06.2025.
2. Algorytmy i struktury danych II USOS UMK, https://usosweb.umk.pl/kontroler.php?_action=katalog2/przedmioty/pokazPrzedmiot&prz_kod=1000-I1ASD-2 ; ostatni dostęp 10.06.2025.
3. Maciej Sysło, *Jak myśleć komputacyjnie*, [w:] *Informatyka w edukacji : myśl komputacyjnie!*, red. Anna Beata Kwiatkowska, Maciej M. Sysło, Toruń, 2018.
4. Załącznik nr 22 do uchwały Nr 41 Senatu UMK z dnia 21 czerwca 2022 r. Program studiów https://archiwum.mat.umk.pl/documents/10180/1282336/inf_I_inz_program.pdf ; ostatni dostęp 10.06.2025.